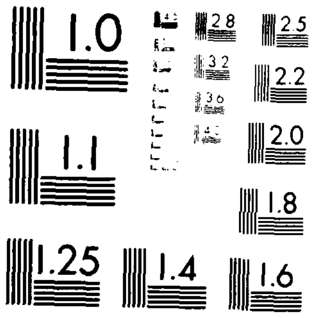
AD-A147 051 THE EFFECTS OF SOFTWARE QUALITY CONTROL AND BASELINE 1/2
MANAGEMENT ON THE AC..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST..
UNCLASSIFIED S C KIMHAN ET AL. SEP 84 F/G 5/1 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

2

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

84 10 31 028

2

AFIT/GLM/LSM/84

THE EFFECTS OF SOFTWARE QUALITY CONTROL AND BASELINE MANAGEMENT ON THE ACQUISITION OF COMPUTER PROGRAMS

THESIS

Sidney C. Kimhan III
Captain, USAF

David M. King
Captain, USAF

AFIT/GLM/LSM/84S-35

AFIT/GLM/LSM/84S-35

THE EFFECTS OF SOFTWARE QUALITY CONTROL AND BASELINE MANAGEMENT ON THE ACQUISITION OF COMPUTER PROGRAMS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Logistics Management

Sidney C. Kimhan III, B.A.
Captain, USAF

David M. King, B.S., M.S.
Captain, USAF

September 1984

Acknowledgements

We sincerely thank our advisor, Mr. William Dean, and our reader Mr. Michael Bates for thei.' guidance and constructive suggestions. Their direction alone enhanced the quality of this report. We appreciate the assistance provided by Mr. Philip S. Babel, Lt Col Robert Carlson, Lt Noah Barr, and Mr. Jeffery Daneman. We also thank Mr. Jim Durlak, Maj Hoblenko, and Maj Kayhua for their encouragement and their presentations which were far more than we expected. Finally, thank you Sandy and Kalei for the moral support.

Table of Contents

List of Figures

List of Tables

AFIT/GLM/LSM/84S-35

Abstract

Cost effective development of quality software for new system acquisitions is an issue of increasing concern within the Department of Defense (DoD). This thesis examines the issue of software development for Embedded Computer Systems (ECS), within the context of the Software Development Life Cycle (SDLC), from the perspectives of Software Quality Assurance (SQA) and Baseline Management. The objective of this research effort is to develop an approach for the joint application of SQA and Baseline Management to improve management control [maintain cost and schedule integrity] during the software development process.

Initially, the disciplines of SQA and Baseline Management are presented as individual components, operating during the SDLC, which provide management with increased control over the software development process. General concepts associated with SQA are first discussed, including the potential role of Software Metrics. This is followed by a review of DoD literature pertaining to SQA and Baseline Management. Having explored SQA and Baseline Management individually, SQA and Baseline Management are then studied as a combined approach towards the effective management control of the software development process.

Through the use of a structured interview, twenty-one Program Managers were surveyed. From the collected cost, schedule and program history data, programs were classified as either having a "Sound" or "Unsound" SQA program and as either "adhering to" or "not adhering to" a baseline management standard. Consequently, analyzing the data using Student's t statistic, the major finding of this research is that there is no statistical evidence that the application of either a "Sound" SQA program or a baseline management standard results in positive cost and schedule control.

THE EFFECTS OF SOFTWARE QUALITY CONTROL AND BASELINE MANAGEMENT ON THE ACQUISITION OF COMPUTER PROGRAMS

I. Introduction

Cost effective software design has become the most significant development problem for new system acquisitions. Over the four generations of computer systems which have evolved in the past three decades, computer hardware has been reduced to a miniaturized scale, achieved very high reliability and low cost, and attained virtually infinite memory capacity. Computer software on the other hand, since it is now being asked to accomplish very complex and sophisticated tasks, has become less reliable and significantly more expensive. Figure 1 typifies the inverse cost trend between expenses for hardware and software. Even more ominous is the observation that three-fourths of the annual expenditures for software are attributable to maintenance activities, the major contributing factor being design (configuration) changes (47:2). Within the Department of Defense (DoD), the increase in software procurement and maintenance dollars has resulted from an increased dependency on computer software for weapons system operation. According to Dr. Edith W. Martin, Deputy Under Secretary of Defense for Research and Advanced Technology,

> Virtually every system in the current and planned inventory makes extensive use of computer technology. Computers control the targeting and flight of missiles, coordinate and control sophis-

> ticated systems within high performance aircraft, and integrate the complex activities of battlefield command. Consequently, software has become the dominant factor in military systems [34:52].

With a USAF software bill now exceeding $2 billion annually, and with increasing dependency on software for national defense, it is not surprising that it is the software element of the computer system which has become a major object of DoD concern.

In a general or theoretical sense, software development involves a series of interrelated, time-phased activities called a development life-cycle (44:2). The Software

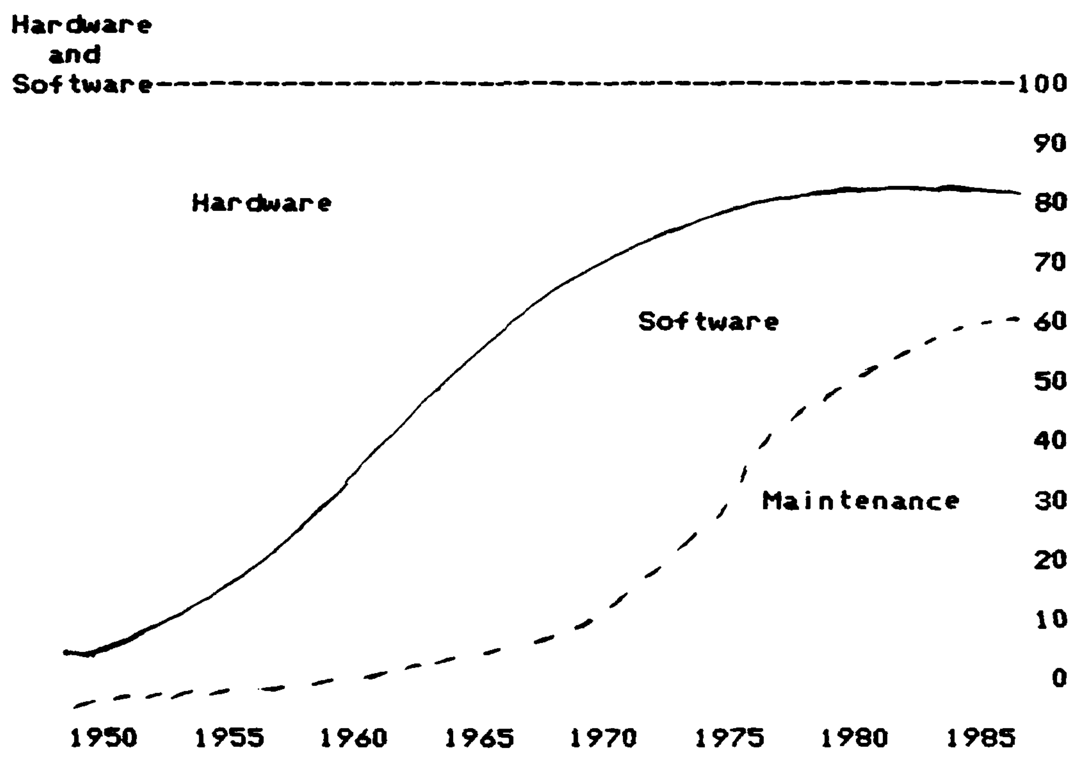


Figure 1. Percentage of Software and Hardware Costs of Total System Acquisition Costs. Source: (40:16).

Development Life Cycle consists of four phases (requirements analysis, design, coding and testing and integration); software's life-cycle is completed in a fif . phase called operations and maintenance (Figure 2). The requirements analysis phase is that phase in the software development life cycle that the role which the software system is to fulfill is defined. In the design phase, software design concepts are explored in an attempt to satisfy system software requirements. The coding phase is the phase in which the actual coding is accomplished. Next, in the test and integration phase, the developed software program is executed to uncover problems that may exist in the code. Lastly, the operations and maintenance phase is that portion

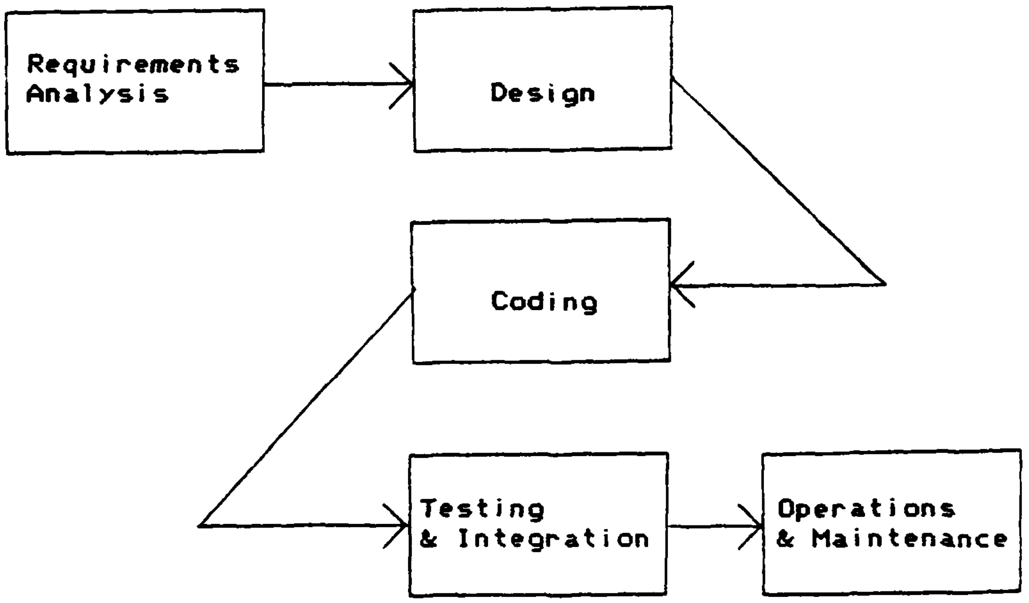


Figure 2. The Software Development and Operations Life Cycle. Source: (29:3)

in the software life cycle in which a software system is turned over to a user (22:67). Inserting checkpoints in this development process enables an assessment of the completeness and adequacy of the software design to be made early in project development, instead of later in the project when recovery may be impossible without adjustments in schedule and cost. The software development life cycle is the conceptualized process of software development (44:240) and software quality assurance accomplishes the inspection of this process (22:217).

Management of software acquisition includes those activities performed to develop software that meets performance requirements while maintaining cost and schedule integrity (6:70-71; 21:13; 23:7). Further, software acquisition management is accomplished through the collective effort of a program management team headed by a Program Manager who is solely responsible for the management of this team (1:20-1). To make efficient and effective decisions, the program manager evaluates information solicited from the other team members who are functional specialists in such areas as quality assurance, configuration management, engineering, logistics, and program control. In the capacity of team leader, the program manager orchestrates the activities of each team member to achieve desired results, addresses how well these results satisfy program objectives, and melds these individual program accomplishments into a cohesive, meaningful collection of information. While administration of each discipline is vital to the overall program success, the emphasis each receives is dictated by the nature of the system being acquired.

Configuration management, however, is one of the most important parts of management control (12:5); without it, chaos results (37:45). Configuration management deserves primary emphasis in all instances, as it will limit or enhance the effectiveness of control that the program manager can achieve over the evolution of the system's design (2:5). Configuration Management is an established discipline which, if properly implemented and kept in effect throughout the program, will help the program manager.

Department of Defense Directive (DoDD) 5010.19 (15:2) *defines configuration management* as the engineering management procedure that includes the following: Configuration Identification; Configuration Control; Configuration Status Accounting; and Configuration Audit. Identification is accomplished by a process known as baseline management; Configuration Control is the management system governing changes made to an established baseline; Status Accounting is the process which provides traceability from the current version of an item or its related documentation to its original baseline; and Configuration Audit is the process used to check an item for compliance with the configuration identification. The importance of the configuration management process is well described by Alton Patterson:

Configuration management is perhaps the least understood, and most difficult to enforce, of all disciplines associated with the management and support of [Embedded Computer System] software, yet without question, it is one of the most important. Many technical personnel, and even managers, because of vague understanding, perceive configuration management as a time consuming overkill control task which obstructs the software change process. Yet, without it, programs invariably get into trouble [41:28].

While all four elements are necessary for the implementation of a sound configuration management program, this thesis will deal exclusively with baseline management to determine the appropriate timing for establishing the functional, allocated and product baselines.

AFR 65-3 defines baseline as: A configuration identification document, or a set of such documents, formally designated and fixed at a specific time during a Configuration Item's life cycle. Baselines, plus approved changes for those baselines, constitute the current approved configuration identification. For configuration management, there are three baselines, as follows:

(a) Functional Baseline. The initial approved functional configuration identification, (b) Allocated Baseline. The initial approved allocated configuration identification, (c) Product Baseline. The initial approved or conditionally approved product configuration identification [19:A-1].

Usually, the functional baseline is documented by an authenticated system specification. The system specification establishes the general performance and functional requirements of the system. The allocated baseline is documented by an authenticated development specification. Development specifications "state the requirements for design or engineering development of a product during the development period [2:17]." The product baseline is documented by an authenticated product specification and referenced detail design documents. Product Specifications describe the "'build to' form, fit, function, and interface requirements and the acceptance tests for these requirements [2:17]."

In this thesis, management is considered to be the degree of effective control that the program manager achieves over the software development. It is measured in terms of how well cost and schedule integrity is maintained and by the degree of quality inherent in the product. Consequently, while configuration management can aid in enhancing schedule and cost integrity during the software development effort, developing a quality software product is also a prime concern, a concern that may be allayed through the incorporation of the expanding science of Software Quality Assurance (SQA).

SQA may be defined as, ". . . a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical standards [43:356]." The underlying premise is that quality is a property of software which is designed in rather than added in later (10:185). SQA attempts to uncover problems early in the software development process and, therefore, significantly reduce the probability of problems appearing later in the program. Gilb concludes that the rule seems to be that the earlier you can find and fix a problem in a software system, the less expensive it will be. The development of software quality assurance programs appear to be the cost effective approach to reducing software development costs and improving software system quality (26:181).

To improve the quality of software, the consensus in the software development industry is:

> The quality of software must be built into the software during the software development period. This implies a need for software engineering rather than the partly organized, non-systematic approaches of the past. Up to now most technological changes have occurred in hardware where classical quality assurance methods applied, but now we are facing a conceptual change in products [22:6].

Software differs radically from hardware (22:8), and although hardware quality assurance practices may apply to software development, in all probability, traditional hardware quality assurance practices must adapt to the nature of software. "Specifically, the concept of built-in quality or 'doing it right the first time' will have to be emphasized [22:8]." A software system product must be designed, fabricated, tested and documented in a disciplined fashion (44:174).

Control is the essential function of a SQA program. "Control is the process of making things happen in conformance with established standards. The basic control process involves establishing standards, measuring performance against these standards, and correcting any deviations from established standards [44:205]." Consequently, SQA is an iterative process that attempts to ensure that each new phase of development may begin only if the preceding phase of work has been performed to acceptable standards. However, for a SQA program to be effective, it must be planned and executed with each task related to a development activity.

According to Reifer (44:264), the objective of configuration management is to control the costs and the reliability of a software system. Further,

> . . . if, under configuration management, specific system requirements were identified at the start of a software development effort; then, as the effort progressed those system requirements could be checked and compared to determine whether those system requirements were being satisfied. Using front end quality planning with the configuration management structure- configuration management could greatly [improve] the software management effort if it incorporated a definition of quality and system objectives [29:17]

Consequently, the configuration management structure can be combined with SQA methodologies within the software development life cycle to significantly improve DoD procured software.

Statement of Problem

As Gansler states, "the most critical issue facing DoD is the increasing use of and dependency on software in weapon systems without the proven management and production methods necessary to control its direct and indirect costs [27:1]." As the complexity and cost of, and dependence on, computer systems continues to increase, better management of computer system development is demanded (3:ii). Consequently, the question becomes one of how can we better manage the development of software to ease the task of maintenance and minimize maintenance attributed to program errors?

Justification of the Research

The work to be accomplished is justified by the major software initiative being presently heralded in the DoD. As Dr. Edith W. Martin said:

> The importance of software to DoD systems is obvious, both in a military and an economic sense. To reap the benefits, however, DoD software must satisfy certain requirements.
>
> The first is reliability. First and foremost, we must ensure that our systems perform the mission for which they are intended, particularly when those missions include life critical situations--reliability is a must. Software must also be adaptable. We must be able to react quickly to changing missions and threats by easily modifying software, sometimes within hours, perhaps even minutes. Finally, software must be affordable. DoD is concerned about the cost of systems. We must find ways to curtail the anticipated burgeoning cost expected as part of the DoD software bill [34:53].

Objective of the Research

The objective of this research effort is to develop a singular approach towards the application of SQA and baseline management to effect management control [maintain schedule and cost integrity] during the software development phase, to minimize unnecessary maintenance requirements caused by design errors [achieving reliability] and to ease enhancement tasks during the operational and support phase of the Embedded Computer System (ECS) life cycle [designing adaptable and flexible software]. Consequently, this research effort will explore how the discipline of Baseline Management may be integrated with developing SQA practices and associated quality assurance tools to ameliorate the management of software development and quality of delivered software products.

Scope of the Research

While software problems experienced in the DoD apply to both Automated Data Processing equipment and Embedded Computer Systems, this thesis is only concerned with the latter. More specifically, this thesis addresses software acquired under AFR-800 series regulations. Additionally, because of limited time,the research will only focus on the software development life cycle. A complete study will require future research into the operations and maintenance phase. However, the work accomplished in the effort will serve as a basis for such future work and will provide important empirical insight as to how successfully the government is currently managing software development.

Research Questions

The following research questions, compatible with the problem statement and the research objectives, guided this research effort:

1. Can the strict application of configuration management, specifically baseline management, enhance the management effectiveness of software development?

2. Can the application of existing software quality assurance methodologies be efficiently applied to develop a higher quality software product?

3. Can a configuration management program and a software quality assurance program be integrated into a single plan which will result in greater benefits than the individual application of either of these methodologies?

Plan of the Report

This thesis has been organized into six chapters. Here in chapter one, the reader has been provided a general account of the problematic history of software development. In this chapter, it was also postulated that the use of SQA and configuration management can result in favorable program outcomes in terms of program cost, schedule integrity, and end product quality. Finally, this chapter was concluded with a problem statement, a justification for research, an explanation of the scope of research and research questions.

Chapters two and three are literature reviews. Chapter two is devoted to a discussion of software quality assurance. This subject is being presented in a separate chapter for two reasons. First, SQA is a new and developing science. Thus, a literature review would provide the reader unfamiliar with SQA with a basic understanding of what the experts consider to be the substance of SQA. Second, the literature review will serve as a foundation for the development of an information gathering instrument, that is, help determine what quantitative and qualitative data must be collected, in order that the test hypotheses developed in chapter four may be tested.

Chapter three is a review of DoD and other government documentation which provide guidance on SQA and configuration management. This review was completed to help understand what, in theory, should be occurring in the management of developing software. This, information will serve as a framework for comparing what should be happening with what is happening in the field.

In chapter four, the methodology used to develop the survey instrument, the structured interview process, and the statistics to be utilized within this thesis effort are presented. Analysis of the data is the subject of chapter five. Finally, the report is concluded with chapter six in which the results are examined. Specifically, an evaluation is made of how well the objectives of the thesis are met, a discussion of what the authors believe the findings indicate, and finally recommendations for future research are made.

II. Software Quality Assurance

Literature Review

> On April 10, 1981, about 20 minutes prior to the scheduled launching of the Space Shuttle, a software fault in the real-time control computer system forced postponement of the first Shuttle orbital flight. Despite thousands of hours of testing and simulation, this hidden fault had not been detected [31:248].

This is only one example of many documented software failures which emphasizes the increasing need for quality computer programs. Further, recognizing that software maintenance costs vastly outstrip the huge initial expenditures associated with software development, the need for quality software becomes even more urgent. With this understanding, this chapter will focus on the concepts which may be incorporated to develop quality software. To begin this chapter, the most recent literature concerning SQA discusses SQA as a two component function, with the first component relating to software quality characteristics and the second component relating to Software Quality Management practices.

The first component of SQA is associated with software quality characteristics or attributes. Poston defines software quality as, "The totality of features and characteristics of a software product that bears an ability to satisfy a given need [43:356]." Consequently, software quality may be viewed as concerned with those characteristics which describe the degree of excellence of computer software (10:7). The idea of software possessing characteristics of quality was initially explored by Ruby and Hartwick (45) in 1968. Ruby and Hartwick identified over 60 candidate software characteristics. In more recent work, Walters, Richards and McCall (35) have reduced proposed candidate characteristics to a set of 11 software characteristics, which are contained in Table I.

TABLE I
Candidate Quality Characteristics
Source: (10:131)

Portability
Reliability
Correctness
Efficiency
Integrity
Usability
Maintainability
Flexibility
Testability
Reuseability
Interoperability

These attributes or characteristics are understood as imputing software quality. For example, portability is defined as the property which allows software to be moved to a new hardware environment with relative ease (28:65). Specifically, if a software product is determined to possess the characteristic of portability, which may be determined through the developing science of Software Metrics, this implies that a software product might more easily and readily be adapted to new uses, which is considered a desirable quality of software. Dunn and Ullman state that, "SQA must contribute to the building-in of these [Table I] attributes in the development of software, and to their retention during the years of software maintenance [22:215]."

The second component of SQA involves the role of monitoring adherence to standards. And, only in the last two to four years has this component become recognized as a formal requirement (10:32). Known as Software Quality Management, "Software Quality Management is the program of planned and systematic activities to determine, achieve, and maintain computer software quality [10:9]." Software Quality Management encompasses the broad spectrum of management activities undertaken to produce quality software. Table II outlines a

TABLE II

A Partial List of Quality Management Activities
Source: (10:10)

1. Preparation of software quality management program plan
2. Development of policies/procedures/standards
3. Software quality assurance audits of documentation, design, configuration management, testing
4. Analysis/evaluation/enforcement
5. Certification/testing
6. Verification/validation
7. Education/training
8. Participation in design reviews and configuration audits
9. Subcontractor control
10. Preservation/handling
11. Program management support
12. Identification and certification of tools, techniques, and methodologies

partial list of Software Quality Management activities. Consequently, software quality management may be viewed as incorporating:

1. Planning. A Software Quality Management plan would serve to summarize all management issues associated with software development. Management issues would include such items as the assignment of responsibilities, timetables for milestone events, and testing procedures. Table III provides a sample format for a Software Quality Management plan.

2. Procedural Assessments. "Software Quality Management encompasses assessment of the procedures and disciplines used in the development, acquisition, management, and maintenance of the software product [10:12]."

3. Product Assessment. This facet of software quality management concerns the review, analysis, verification, and validation of the software product.

Therefore, Software Quality Management is concerned with the procedures followed in the development of software and makes up the second component of the SQA function.

Understanding that SQA is a two component function, an SQA program incorporates and fosters the development of both software quality characteristics and software quality management activities in an attempt to develop quality software. The difference between the software quality characteristic component and the software quality management component of the SQA function is in the type of control exercised in the software development process. Specifically, software quality characteristics imply control through the establishment and measurement of the characteristics themselves; software quality management practices imply control through the specification of milestones and procedures (10:11). Consequently, with each component exercising control over a different facet of the software development effort the goal of designing quality into the software product may be realized.

TABLE III

Format for a Software
Quality Management Plan
Source: (10:12)

1.0	Management Overview
	1.1 Objectives of plan
	1.2 Schedule
	1.3 System overview
	1.4 Management control procedures
	1.5 Organization/resources
2.0	System Functional Summary
	2.1 Information required
	2.2 Software development process
3.0	Software Quality Requirements
	3.1 Software quality factors
	3.2 Software quality metrics
4.0	Life Cycle Tasks
	4.1 Software quality management flowchart
	4.2 Task descriptions
5.0	Documentation Requirements

Software Quality Characteristics

With regard to controlling software quality characteristics in an SQA program, ". . . as a general rule, the system designer must identify all essential system qualities [28:67]." Furthering this premise, Miller and Howden have shown that the degree of quality a person puts into a program correlates strongly with the software quality objectives given (38:29). Gilb also concludes that the more clearly a goal is stated, the more likely it is that programmers will try to meet that goal (28:67). Therefore, from the literature reviewed, the need to precisely specify desirable characteristics within software is essential.

To assist in this task, the newly developing science of Software Metrics is providing a controlling framework within which to more concisely define software characteristics. "Software metrics is a new area of science aimed at assigning quantitative indices of merit to software [42:1]." Software metrics provides the measurable goal against which the quality of software can be assessed. Metrics provide quantifiable measures for software quality by measuring the degree to which a software product possesses and exhibits a certain characteristic (38:290). In Michael Cooks article, "Software Metrics: An Introduction and Annotated Bibliography," the concept of software metrics is introduced. Software metrics is viewed as a developing science which will provide quantitative measures of software characteristics throughout the development of a software product. The purpose of software metrics is ". . . to see how they reflect the quality of what is being measured. Software metrics can be used to evaluate the effectiveness of programming methods and the reliability of a system [9:41]." In concluding, Cook states, ". . . some metrics are easy to apply and useful, while others are difficult to implement and costly. The foundations of software measurement are still being laid and software metrics will be an important area of research [9:43]."

While software metrics presents a promising opportunity to significantly improve software quality, skepticism exists in the scientific community concerning the employment of the scientific method in the development of metrics. In a recent paper by Johnson, the significant role which software metrics could fulfill within SQA programs is recognized. However, Johnson is skeptical claiming that although many metrics have been developed, ". . . many have yet to be demonstrated and validated for actual use [30:184]." Maintaining a similar view, Perlis, Sayward, and Shaw, editing a 1981 publication entitled Software Metrics, pose the question, "Can there be assigned to software and the processes associated with its design, development, use, maintenance, and evolution, indices of merit that can support quantitative comparisons and evaluation of software [42:preface]?" While Dunn and Ullman provide an excellent work on software development (22), these authors strongly question the utility of software metrics (22:96). Denicoff and Grafton summarize the challenge put forth to software metrics researchers, "A significant challenge to software metrics researchers is to stay within the traditional paradigm of hypothesis, evaluation, criticism and review [14:205]."

Despite skepticism, software metrics research is being pursued. Murine reports promising results when software metrics were included in a SQA program on a major defense system software project. Murine concludes, "We have discovered that incorporation of the Software Quality Metrics [Software Metrics] methodology into an SQA program has satisfied all our initial quality objectives as well as some not previously contemplated. It provides a real, positive quality impact on software product development and measurement and can be used as a major cos treduction tool as well [39:188]."

Software Quality Management

However, it should be recalled that software attribute control represents only half of the control process within an SQA program. Software Quality Management practices make up the remaining half of an SQA program. Chow summarizes the basic activities or practices associated with software quality management concluding that software quality management consists of three elements (8:351):

1. an SQA plan,
2. quality control techniques, and
3. tools.

Poston expands and elaborates on Chow's basic elements of software quality management. Poston concludes that a soft ware quality management program includes (43:356):

1. Policies. A set of policies that would guide the implementation and application of the SQA program.

2. Methodologies. This refers to clearly defining and documenting the software development process throughout the software development life cycle.

3. An SQA Plan. This plan would provide a written record of all SQA activities including what part of the organization is responsible for which activities.

4. Standards. The selection and creation of standards for documents, test coverages, configuration management reports and other documentation as outlined in the SQA plan.

5. Tools. Tools to assist in the software development process by providing the mechanism to ac-complish standards.

6. Reviews and Audits. Provide the opportunity to monitor the software development process.

Poston's elements present a theoretical framework for the development of a software quality management program within which software quality management practices may be applied. Further, in addition to the above mentioned elements, there exist four support factors which also contribute to enhance software quality.

The first support factor which contributes to the development of quality software is management support. Specifically, full support of management must exist for software quality goals to be realized (10:64). Dunn and Ullman note that it is probably a mistake for management to attempt to impose a complete quality system all at once (22:253). However, a formal SQA program would indicate a commitment by management to the requirements for a quality operation, in addition to making SQA a more visible and identifiable function (29:7). The full support of top management is viewed as the first key support factor in the development of quality software.

The second support factor derived from the literature reviewed implies that ". . . the SQA program should be made an independent effort by a functionally separate team [48:7]." Demarco states that the advantages of an independent team include (13:55):

1. estimators having no emotional stake in the project, therefore, estimators are relatively free from pressures to come up with 'the right answer,'

2. estimators learning through substantial repetition, and,

3. a centrally controlled data collection that will result in homogeneous measurement.

While these advantages may seem obvious, the real advantage is the management visibility provided by a team lacking any conflict of interest for project completion. The establishment of an independent evaluation team is the second key support factor which, as indicated by the literature reviewed, significantly contributes to "designing-in" quality.

The third support factor to be highlighted relates to the testing of software. Testing implies (44:211):

1. the establishment of predetermined goals or standards,

2. the measurement of performance, and,

3. the comparison of actual performance with standards.

However, while the implication may appear to be that of assessing software performance, experts conclude that

". . . the primary goal of testing should not be the demonstration of correct performance but the exposure of hidden defects [22:180]." This perspective is substantiated from previous emphasis on the fact that the longer a defect remains in the system, the more expensive it is to remove. The testing support factor is operationalized through the use of software testing tools. The understanding is that ". . . the software development process has a greater chance of success if the instruments for measuring the development process are sufficiently detailed and accurate and used at a sufficiently early point in the work process [28:50]." Table IV contains a partial list of software testing tools. Testing is the third key support factor and is viewed in the literature as imperative in the software development process.

The final key support factor identified in the literature as contributing to the software development process relates to software development progression. As discussed previously, configuration management establishes a series of control reviews. With respect to a software quality management program, work should only be allowed to progress when all specified review criteria have been fulfilled (10:xiii; 22:219). The SQA plan must decompose the whole software development process into stages. At the end of each stage, a control point must be defined with a complete specification of the expected product and the quality criteria for exiting one stage and entering the next stage. Quality assurance methods must be applied at each control point to assess the quality of the product (8:351). The idea is that, ". . . an SQA program must include the provision that work may proceed only with the concurrence of software quality assurance, as based on the review of previous work [22:219]." Insuring software progression criteria are met combined with the previously mentioned key support factors of testing, independent inspection and management support will insure a higher quality software product.

TABLE IV

Software Testing Tools
Source: (22:186)

Basic diagnostics
Change tracker
Comparator
Definition and design processor
Dynamic analysis
Emulator and simulator
Flowcharter
Global cross-reference mapper
Host system
Librarian
Link editor
Postprocessors
Preprocessors
Regression test system
Software development workbooks
Standards analyzer
Static analysis
System performance monitor
Test case generator

Software Quality Management serves as the mechanism for control of Software Quality Management practices within an SQA program. In an analysis of the literature to date, the consensus by industry experts focuses on an effective software quality management program as incorporating such elements as previously described by Poston and the four key support factors identified above. Software Quality Management is the second component of an SQA program and further insures that quality is built into a software product from the beginning instead of attempting to add it at the end of product development as has traditionally been the case.

Summary

In the most recent literature an effective SQA program is viewed as a two component function. The first component concerns the development of various software characteristics, which are defined and measured within the developing science of Software Metrics. The second component concerns various Software Quality Management practices. Consequently, it is concluded that when efforts are made to "design-in" quality characteristics within a structured framework of software quality management practices, the end result will be a higher quality software product. However, given that the basic concepts have been defined to support a control methodology for an effective SQA program, an anomaly exists in that reports of low quality software abound.

Continued reports of high software development costs and low user satisfaction with procured software emphasizes the need for developing quality software. Consequently, this chapter has focused on the general concepts contributing to the development of quality software for the two-fold purpose of, first, providing the reader with a basic understanding of current beliefs and methodologies purported to contribute to the development of quality software and, second, serving as the foundation for a data gathering instrument, which will eventually be used to accept or reject thesis hypotheses. In the next chapter, DoD SQA will be reviewed, together with baseline management control methodologies, in an effort to discern DoD practices for obtaining quality software.

III. Government Guidance on Configuration Management and Software Quality Assurance

Introduction

Literature devoted to software management control abound. The majority of these expositions can be divided into two categories. The first, because of the exorbitant costs being incurred, emphasizes the importance of effective management and control during the development of the software product. Such material was used to develop the introductory chapter and is further cited throughout this report. The second category addresses the different approaches or management techniques that may be implemented to better manage software development projects. This second category is the emphasis of this chapter.

Baseline Management

One technique which is repeatedly identified as a strong management control tool is Configuration Management. Only a modicum of the literature, however, deals with the application of configuration management to a defined development process. DoD directives, and other related government documents are perhaps the most fertile and relevant source of data providing such direction. As Anway observes, ". . . the directives within the DoD that address configuration management are written for the acquisition process of major defense systems [3:1]." Table V lists the six major government documents, pertinent to Air Force acquisitions, which deal configuration management. Examination of these documents reveals, however, that caution must be exercised when relying on these directives, as ambiguities and contradictions between related documents risk broad or incorrect interpretation. Through an analysis of these six documents, the remainder of this section will include (1) a definition of baseline management, (2) a discussion of document precedence within the Air Force, (3) an examination of the discrepancies among these documents, and (4) an indication that AFR 800-14 should be considered the proper guidance document for acquisition of software within the Air Force.

TABLE V

Government Documents on Configuration Management

DoDD 5010.19
AFR 65-3
AFR 800-14
MIL-STD-483
MIL-STD-490
MIL-STD-1521

Air Force MIL-STD-483 describes baseline management as "one of the more important aspects of configuration management . . . which is formally required at the beginning of an acquisition program [16:4]. MIL-STD-483 continues:

> Baselines may be established at any point in a program where it is necessary to define a formal departure point for control of future changes in

performance and design. System program management normally employs three baselines for the validation and acquisition of systems to include the functional, allocated, and product baselines. . .
Baselines are the basic requirements from which contract costs are determined [16:5].

DoD Directive (DoDD) 5010.19 is the government source document which establishes top level guidance for configuration management. Within the Air Force, AFR 65-3, Configuration Management, which implements DoDD 5010.19, is the authoritative source from which policy on timing for software baseline control is obtained. According to AFR 65-3 (19:2-1-2-2):

> . . . the functional baseline, will be a product of the conceptual effort. . . the allocated baseline, will be formally established during Advanced Development/Va 'ation or Full Scale Development . . . the prouuct baseline, shall be established upon successful completion of a Physical Configuration Audit.

Baseline control timing is the same in AFR 800-14 (20:2-1-2-2) as it applies this policy specifically to software acquisition. The authenticated (baselined) system (functional) specification is the definitive document resulting from the Conceptual phase. At the beginning of Full Scale Development (FSD), and prior to Preliminary Design Review, the development specifications should be completed and authenticated. It is during the Production phase that control of the product baseline is established. AFR 800-14 contains the most specific software policy on when each of the three identifying baselines should be established and is explicit as to what software related activities, information and activities, information and documentation must be available before the government takes control.

MIL-STD-483 also attempts to establish general time frames for appropriate baseline points in the acquisition process. According the MIL-STD-483:

> The timing of the establishment of the functional baseline will be as agreed between the contractor and the procuring activity, but not later than Preliminary Design Review . . . The allocated baseline will be formally established with the award of engineering or operational systems development contract(s) whenever possible. The timing of the establishment of the allocated baseline will be as agreed between the contractor and the procuring activity, but not later than CDR [16:7].

MIL-STD-483 does not specify when product baseline should be established.

In contrast to MIL-STD-483, which is concerned specifically with con figuration management practices, MIL-STD-1521 outlines conduct for the different design reviews and audits. In defining the purpose and objective of each review and audit, it provides more definitive timing for baseline control actions. According to MIL-STD-1521 (18:12,15,18,30), functional baseline should be taken at the end of the conceptual phase or beginning of the validation phase; allocated baseline should be established well before Preliminary Design Review (PDR); and product baseline should be delayed on into the production/operational phase after Physical Configuration Audit (PCA) is completed.

AFR 65-3 states that, "The configuration management process shall be carefully tailored to the . . . nature . . . of the CI involved [19:1-1]." The intent of this verbage has often been misinterpreted and resulted in varying implementation from one program to another. Within the Air Force, it should be understood that AFR 800-14 has tailored the process for ECS and should be the standard followed for a software acquisition program.

Like AFR 65-3, MIL-STD-1521's authority is also jeopardized. The source is from a "note" within that document which reads: "Actual time phasing of activities must be tailored to each program [18:10]." MIL-STD-483 is the most guilty of the discussed sources which frustrates the issue of when to establish the three baselines. As may have been noted from above, MIL-STD-483 allows functional baseline to be delayed up through PDR and allocated baseline to be established as late as Critical Design Review. Additionally, another source of weakness in MIL-STD-483, like MIL-STD-1521, stems from a" note" which reads:

> On some programs, particularly major programs, forcing establishment of allocated baseline as early as 90 days after award artificially restricts design solutions causing costly changes downstream. The agreement should be documented e.g., in the configuration management plan, if one is used on the program/project [16:7].

The flexibility built into these documents are perhaps workarounds made available to account for the many circumstances that exist in the acquisition of a product. However, as the literature indicates, there exists a natural order of events which occurs in all acquisition programs. Therefore, a program manager should not be willing to proceed from one event to the next without being satisfied that required documentation is available and related activities are accomp lished.

It is true that a program may not have had Conceptual or Validation phase. However, even if its inception is at the beginning of FSD, a top level (system/prime item) specification is still needed; development specifications are based upon the allocation of requirements from the top level specifications; and product specifications are a complement to their associated development specification. This sequence, and its relations to program engineering design reviews, is set in AFR 800-14, which governs the acquisition of embedded computer resources. Therefore, determination for baseline control subsequently used in the remainder of this thesis shall be based on the discussions set forth in AFR 800-14.

Software Quality Assurance

Within the DoD, quality assurance is viewed as the, ". . . planned and systematic pattern of all actions necessary to provide adequate confidence that material, data, supplies and services conform to established technical requirements and achieve satisfactory performance [17:1]." With reference to SQA, Table VI outlines the DoD documents which reference software quality.

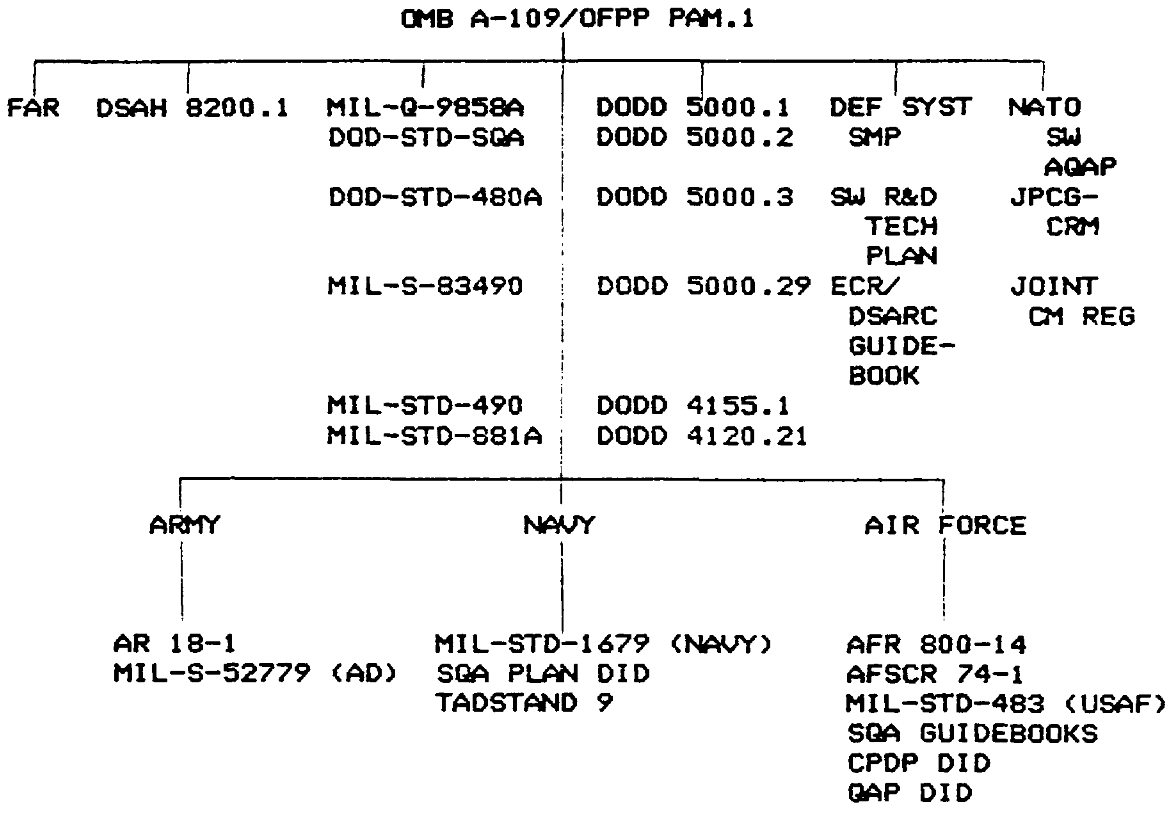
DoD initiatives to improve the quality of software obtained began in the mid 1970's. Prompted by problems which still exist in the software industry today, the government began a review of how systems were acquired. Known as the Acquisition Improvement Program, the focus was on increased acquisition effectiveness and efficiency. As a result of these efforts to improve the Acquisition process, DoDD 5000.1 was published (32:1). The policy and management principles applicable to major system acquisitions are established in this directive. Specifically, DoDD 5000.1 outlines the responsibilities of top level managers and key decision points in the system acquisition process. However, "DoDD 5000.1 doesn't address computer resources (hardware, software, personnel, facilities, etc.) or software specifically . . . [32:1]." Consequently, while DoDD 5000.1 was the initial attempt to improve system acquisition, it was left to later documents to recognize the significance of software in the system's acquisition process.

One of the first top level DoD documents to specifically recognize the significance associated with software is DoDD 5000.2. This directive emphasizes that plans for software development, documentation, testing and updating during software development require special attention (32:17). DoDD 5000.2 establishes the policies and procedures, including policy guidance for the acquisition of computer resources, for DoD activities in support of DoDD 5000.1.

DoDD 5000.29 is also an upper level management document which should be considered. This document establishes the DoD policy for the management and control of computer re-

TABLE VI

DoD Software Quality Documents
Source: (10:102)



sources with increased emphasis on software. DoDD 5000.29 establishes policy for the management and control of computer resources during the development, acquisition, deployment and support of major defense systems. The provisions of this document encompass major programs as designated by DoDD 5000.1, but its principles are also to be applied in the acquisition of defense systems that do not fall in the 'major acquisition category' [21:27]. In addition, DoDD 5000.29 specifies milestones and the need to associate specific criteria to determine milestone attainment through the Software Development Life Cycle. DoDD 5000.29 provides policy guidance for the management of computer resources.

The various 5000 series directives discussed in this paper indicate an upper level (DoD) managerial commitment to developing quality software; the Air Force 800 series regulations implement the 5000 series policy. Specifically, AFR 800-14 deals with the acquisition and support for computer resources in weapon systems; it is the implementation of DoD directives 5000.1, 5000.2 and 5000.29. AFR 800-14 is the first in the series of Air Force (AF) acquisition or management-oriented regulations to specifically address software (21:28). "The primary AF policy on the management of computer re ources in weapon systems is contained in AFR 800-14 [32·1]."

AFR 800-14 is a two volume publication. The first volume of AFR 800-14 focuses on top level functional management support for computer resources and systems. The purpose of this volume is stated as:

> . . . insure that computer resources in systems are planned, developed, acquired, employed, and supported to effectively, efficiently and economically accomplish Air Force assigned missions [20:3].

This volume emphasizes the importance of the correct definition of software requirements.

The second volume of AFR 800-14, ". . . is meant to be a definitive treatment of the policies, procedures and guidance required to manage computer resource development, as defined by AFR 800-14, Vol I [32:57]." This volume covers a broad spectrum of management topics from the planning and engineering of computer resources to documentation, testing and configuration management as related to computer resources. In addition, AFR 800-14 is of special interest since this regulation serves as the source document for the establishment of an SQA program during software development. Specifically, AFR 800-14 outlines the requirement for a Computer Program Development Plan (CPDP). The CPDP states the steps required to complete a project and the methods that will be followed in each step (22:216). Specifically, the CPDP is concerned with the controls required to generate quality software.

> The CPDP is considered to be a management plan and, hence, a living, adaptable document which defines objectives, breaks down work tasks, assigns work schedules and institutes the monitoring required to feed back progress estimates [32:8].

It is of interest to note that AFR 800-14 does not specifically address SQA but instead leaves SQA to be dealt with through the CPDP and supporting Military Standards. AFR 800-14 is concerned with how the computer resource acquisition process should be implemented.

In support of AFR 800-14, there exist several documents which are often imposed on contractual requirements. MIL-

STD-483 establishes uniform configuration management practices and specifications which can be applied to all USAF systems, equipment, munitions and computer programs. MIL-STD-490 describes, "the purposes, formats and technical content of various types of specifications . . . [32:49]." And, MIL-STD-1521A describes the requirements for conducting the various milestone events. "This standard outlines the minimum information required for each type of review and audit [25:177]." Although these documents do not specifically detail SQA programs, the need for SQA is recognized through requirements for testing. One criticism levied on MIL-STDs 483 and 490 is that, while these documents do indicate milestones and performance testing, the actual work to be accomplished and products to be delivered are not defined (21:25). However, according to Driscoll, MIL-STD-1521A does define work and product deliverables.

A more recent standard which deals exclusively with SQA is MIL-S-52779A, "Software Quality Assurance Program Require ments." "MIL-S-52779A establishes the requirement for a definitive, visible contractor SQA program and associated planning documents [32:48]." This specification addresses such topics as:

1. Tools, techniques and methodologies. Concerns whether or not the contractor identified and defined the software engineering tools, techniques and methodologies planned to support the requirements of SQA.

2. Computer program design. Concerns how the SQA program will handle procedures for the review and evaluation of software design documentation.

3. Work certification. Concerns the contractor's procedures for formally approving the completion of work performed under the contract.

4. Documentation. Concerns how the SQA program shall insure compliance with accepted standards, practices and conventions of documentation during software development.

5. Testing. Concerns the process of explicitly assuring that the software performs as required.

Consequently, MIL-S-52779A, ". . . has generated an increased awareness of the need for SQA in software development [48:1]."

MIL-S-52779A is a brief document the contents of which are further explained in Military Handbook (MIL-HDBK) 334. MIL-HDBK-334 was published to clarify MIL-S-52779A and to, ". . . provide guidance for the evaluation of a contractors SQA program when MIL-S-52779A is invoked [32:iv]." This handbook emphasizes the planning and execution of a comprehensive SQA program. In addition, the handbook specifies that a contractor's SQA program should be developed to identify deficiencies early in the development process. MIL-HDBK-334 also notes the strong relationship between configuration management and SQA. This handbook attempts to clarify issues created by MIL-S-52779A.

However, while the DoD has attempted to improve software quality through software quality management practices as contained in the above directives, regulations, standards, and specifications, low quality software continues to be developed. Low quality software is characterized by "cost and schedule overruns, poor performance of systems when delivered, high maintenance costs, lack of reliability, and a high degree of system sensitivity to changes in requirements [29:13]." Consequently, it is argued that while DoD documents allow for the existence of an SQA program, an SQA program in and of itself is not enough to insure the development of quality software.

Summary

According to the literature reviewed, timing for baseline control should be established in relation to engineering design reviews as follows. A functional baseline should be established as a result of a Systems Requirements Review (SRR). The functional baseline forms the basis of the System Design Review (SDR). After the conclusion of the SDR, allocated baselines should be defined and established. A PDR follows the establishment of the allocated baseline which then serves as the foundation for the detailed design work. Draft product baseline documentation should be completed ahead of a CDR. After CDR, the detail design is tested and the draft product baseline documentation is refined and completed. Formal establishment of the product baseline should be rendered after Physical Configuration Audit (PCA) has been conducted.

Within the scheme of this management plan, a strong commitment to establishing these baselines at these specific points in the program is essential. Further, the program manager must be unwilling to progress from one baseline to the next until necessary documentation of the proper form and content is available. This last qualification implies a notion of quality being designed into the product as it matures from beginning to end.

As discussed in chapter two, the concept of quality as traditionally applied to the development of a hardware product, ". . . easily lends itself to quantifiable standards [22:217]." However, with respect to the development of software, "The science of managing software development is still in its infancy, and the lack of a good clear set of principles is apparent [44:251]." Nevertheless, in researching the topic of quality, as related to the development of a software product, a single principle did emerge as significantly contributing to the development of a quality software product. Specifically, the idea of "designing-in" quality throughout the course of the Software Development Life Cycle appeared as the key to developing quality software. Consequently, the end result supports a conclusion that, for the development of a quality software product, quality must be designed into a software product through a formalized set of controls- controls which are provided by the discipline of configuration management.

"Configuration Management provides the detailed framework upon which a SQA program can be built because it establishes the formal structure necessary to enforce compliance with procedures [44:9]." Specifically, through a series of engineering reviews and configuration audits technical and administrative direction are applied to (a) identify and document functional and physical characteristics of systems and configuration items; (b) control changes to those charac teristics; and (c) record and report change processing and implementation status (20:6-1). Consequently, this direction allows for implementation of the essential SQA support factors-- of securing management support for the development of quality software, formulating an independent SQA team to ensure the development of quality software, conducting suffi cient testing to verify software quality, and ensuring the next phase of the development process is not entered into until existing phase criteria are fulfilled. Therefore, in the opinion of these authors, a particularly strong relationship exists between the disciplines of SQA and baseline management. A relationship which is not utilized to its fullest potential.

IV. Methodology

Introduction

This chapter summarizes the methodology used in the collection and analysis of data necessary to answer the thesis' management question. Specifically, the concerns to be addressed include (1) a discussion of the procedural design of the research, as developed, to yield the most objective results; (2) a discussion of reporting procedures, with emphasis on the objectivity of reporting; and (3) validation of the survey instrument. Further, this chapter discusses what was expected to be found; the statistical tests chosen to test data; and the assumptions made in formulating the overall design of this research effort. This chapter provides a detailed outline of the research procedures to allow a fair replication of this research.

This thesis' primary assertion is that a software quali ty assurance (SQA) plan together with baseline management can be integrated into a single management program whereby a software acquisition development program's cost and schedule integrity can be effectively maintained and the computer program's reliability enhanced, thereby reducing its maintenance requirements during the operational phase of its life. This premise, however, presumes that both SQA and baseline management have some level of influence over the management control of the product. Therefore, in exploration of this premise, it is first necessary to determine if

SQA and baseline management independently contribute to program integrity. If SQA and baseline management do independently influence program integrity, this information can then be used to probe the possible joint effect of SQA and baseline management upon program integrity.

Developing the Data Gathering Instrument

The design of a longitudinal experiment allows for an investigator to observe the reaction of a test group to varying stimuli in comparison to an unexposed control group. In consideration of the time allowed to conduct this research and the nature of the situation to be explored, it was necessary to find an alternative means to collect data other than a form of a longitudinal study. Consequently, operating within a limited time dimension and with no control over the variables involved, it was decided to develop this research effort as a cross-sectional field study within the context of an ex post facto design. Further, understanding that this research may be classified as primarily causal in nature, it was necessary to attempt to uncover comparative groups which have and have not been exposed to the "causal" factors under study.

The chosen course was to obtain and evaluate historical data through the use of a questionnaire as administered through the process of a structured interview. Using the hypotheses as a guide for the direction of this study, the questionnaire was designed and constructed to obtain needed data. The design, construction and use of the questionnaire was based on several considerations.

First, the criticisms attributed to survey methodologies were reviewed. Specifically, while complete dependence of verbal responses may result in untrue or misleading answers, the versatility associated with an interrogation process outweighed any shortcomings of this method. The personal interview was viewed as the most practical and economical way to uncover needed information.

Second, a one-on-one interviewer-interviewee relationship was pursued to take advantage of the greater depth and detail of information that could be secured, as well as the possibility of gathering higher quality information through noting the conditions of the interview, additional probing, or general observation.

Third, the questionnaire was developed in a standardized format and sequence to help assure that each question was asked in the same way, thus promoting measurement reliability. As a corollary to the question structure, the response structure afforded the interviewee was also considered. Consequently, while many questions could be answered yes or no, because of the complexity of the subject matter respondents were given the opportunity to elaborate and provide qualifications to their answers. Therefore, a more thorough understanding of any explanation could be gained while minimizing risk of any misunderstanding.

Lastly, objectives of this research effort were not disguised, since respondents were knowledgeable at a conscious level of needed information and willing to provide information. In addition to these considerations, it was also felt that a questionnaire used in conjunction with a structured interview would provide increased continuity in the data gathered among the different programs investigated.

Sample Population

This research effort was confined to programs managed at Aeronautical Systems Division (ASD) in which system or subsystem software was being or had been developed. The scope of this research was limited to system software for the purposes of standardization, and only ASD programs were selected to prevent possible deviations in the data collected due to local policies and practices potentially inherent in other Air Force System Command product divisions. Initial introduction to the population of ASD acquisition programs was obtained through a computerized list maintained by AFALC/XR, which identified Deputy Program Managers for Logis tics (DPMLs) and Integrated Logistic Support Managers (ILSMs) supporting 93 ongoing program efforts. The DPML/ILSM listing was deemed to be a convenient means of identifying program managers. DPMLs or ILSMs were contacted and were requested to identify ASD program managers currently managing software development programs. ASD program managers were also identified during the actual interviewing process when respondents were asked to identify other programs which may meet the criteria of this research effort. For programs currently developing software, this research effort dictated that the program be in the later stages of Full Scale Development, so that a sufficient cost and schedule history would be available to correlate to the existing SQA and baseline control programs.

Since later candidate interviews were obtained through suggestions of interviewed program managers, a concern arose that the sample's randomness would be impaired. However, according to Mr. Jeff Daneman, AFIT/LSQ, who approved of the sampling methodology, while many texts consider randomness to mean that each element of a population has an equal chance of selection, the concept of randomness also means that there is no discrimination in selection (11). Therefore, because every referenced program was contacted and each contacted program manager who was willing to participate was interviewed, the sampling was completed without bias and can be considered random.

Planning for Control of Situational Factors

An additional concern to this research effort was the actual interrogation process itself. Viewed as critical to the success of this research were Emory's broad criteria for a successful personal interview. Emory's criteria are (24:294):

1. accessibility by the respondents to needed information,

2. an understanding by the respondents of their roles, and

3. motivation of the respondents to accept such a role and to fulfill its requirements.

Prescreening of the respondents prior to arranging an interview appointment insured the accessibility of respondents to needed information. At the beginning of each interview, every effort was made to gain a good interviewing relationship. Research objectives were explained in an attempt to outline the role of the respondent and to motivate the respondent towards an understanding of the importance of this study. Additionally, in terms of actually conducting the interview, all interview appointments were scheduled at the program managers convenience, and it was emphasized that the interview was to take a limited amount of time and that any interruptions would be understood. During the interview, each question was read directly from the questionnaire and then clarified with respect to intent, as needed. Responses were recorded during the interview by both researchers for purposes of accuracy, and, after each interview session, recorded responses were compared to identify and clarify any recorded deviations.

Planning for Measurement

Measurement is the process through which the hypotheses in this thesis will be tested. However, before any measurement could be undertaken with the questionnaire, the questionnaire had to be validated. Thorndike and Hagan state that validity, "...refers to the extent to which a test measures what we actually wish it to measure (24:128; 46:5)." While many methods exist to validate an instrument, the validation of the survey instrument utilized in this research effort was accomplished through expert evaluation. Specifically, both the data gathering instrument and data gathering approach were discussed with experienced individuals within the ASD community (4; 5; 7; 11) respected for their knowledge of software development and statistics.

To test the hypotheses, the collected information, needed to include a description of a program in terms of its (1) SQA program, (2) baseline management policy, and (3) cost and schedule performance track. With such information, the plan was to substantiate that if a program had both a sound SQA program and complied with the baseline management standard, then the program would reflect a favorable cost and schedule history. Conversely, a program which did not have a sound SQA program or employ the baseline management standard would suffer from significant cost overruns and schedule slips. While concrete examples of both situations would provide the cleanest and easiest data to handle, the reviewed literature suggested that no such clear-cut situation exists. Consequently, to ensure that all programs would be treated equally, a singular method to handle data was established prior to initiation of the interviews as is subsequently discussed.

<u>Software Quality Assurance Program</u>. The SQA literature

reviewed in chapter's two and three concerned both non-DoD and DoD attempts to insure the development of quality software. Specifically, chapter two emphasized the concepts that serve as the foundation for an effective SQA program, while chapter three reviewed DoD attempts, through various directives, regulations, military standards, and military specifications, to operationalize chapter two concepts within the DoD. Consequently, recalling that the primary concern associated with the SQA hypothesis requires a determination of the soundness of an SQA program, the generic concepts discussed in chapter two will serve as the primary source for this determination rather than the specific requirement addressed in chapter three.

Programs based upon the concepts presented in chapter two were considered to have more effective (sound) SQA programs and would lead to the eventual production of quality software. Specifically, the four key support factors discussed in chapter two were viewed as contributing to an effective SQA program. The first support factor was the degree of upper level management support by the contractor producing the software and by the program office procuring the software. The determination of such support was made based on the existence or non-existence of upper level management policies, methodologies, SQA planning documents, standards, reviews and audits, and the degree of commitment to such items. The second key support factor concerned the establishment of an independent SQA activity. Within a program office, an independent SQA activity usually takes the form of an Independent Validation and Verification team, which is a group of either Air Force engineers or privately contracted software engineers attempting to verify software performance. Within a contractors facility, an independent SQA activity was considered to be an independent function specifically devoted to SQA. The third key support factor concerned testing. The testing aspect of developing quality software dealt with how the testing process was conducted, who was present at testing and how discrepancies were documented and tracked through to completion. The final support factor explored concerned the controlled progression of software development, where the position taken was that no new phase of software development should be undertaken until work in preceding phases was accomplished to the satisfaction of the program manager.

Consequently, from these support factors, which were incorporated in the questionnaire, the determination of the effectiveness or non-effectiveness of an SQA program was made. As will be explained in chapter five, all programs not founded in the above mentioned support factors were considered to have an unacceptable (unsound) SQA program. The term unacceptable only means that the program SQA plan did not appear to be based in the support factors espoused within the literature reviewed.

<u>Baseline Control</u>. As discussed in chapter three, AFR

800-14 was the primary source from which the appropriate timing for establishment of the respective specification baselines (functional, allocated, and product) was determined. Specifically, if a program had an established functional baseline after SRR but prior to SDR, an established allocated baseline prior to preliminary design review, and an established product baseline with the completion of the physical configuration audit, the program was considered to have complied with the baseline management standard. All other instances were considered to deviate from the baseline management standard.

Significant Cost Overruns. The cost hypothesis suggests that a significant cost overrun will occur when an unacceptable SQA or baseline management policy is employed by a program. Each interviewee was asked what they consider a significant cost overrun to be on a program, as a percentage of the program cost. A significant cost overrun will then be determined by averaging the responses of interviewed program managers.

Actual Program Cost Overruns. The actual program cost overrun will be calculated as the cost of engineering change proposals (ECP) due to software deficiencies (excluding identified deficiencies because of new requirements) divided by the cost of software in the program. Thus, a $300K software ECP on a $1M program will be equivalent to a $30K software ECP on a $100K program; each equalling 30 percent.

When the mean cost overrun is being calculated, the cost overrun percentage will be multiplied by a basis of $100. For example, if program XYZ's cost overrun percentage is 15 percent, then program XYZ's cost overrun will be calculated as, $100 X .15, or $15.

Significant Schedule Slip. In each interview, the program manager was asked to specify their understanding of a significant schedule slip. An average of the responses will be used as the decision rule to determine if a program does or does not have a significant schedule slip. Based on the preceding decision rules the cost and schedule information can be classified into one of two categories for each of the areas being studied.

Statistical Test

For the type of information expected to be gathered, an appropriate statistical measure is the one-tailed t test for small sample inferences about the difference between two population means. A small sample implies a sample size of less than 30 observations for each population being sampled. The key assumption is that, "To use the t statistic, both sampled populations must be approximately normally distributed with equal variances, and the random samples must be selected independently of each other [36:337]." This assumption is made based on the understanding that the sampling distribution of the sample means, x and x , depends on the shape of the populations being sampled. The use of the t statistic involves the proposal of a null hypothesis, which is either accepted or rejected in favor of an alternative hypothesis based on a selected level of significance. The t statistic is of the form

$$t = \frac{x_1 - x_2}{\sqrt{s_p^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

$\bar{x}_1$ = sample mean from sample 1
$\bar{x}_2$ = sample mean form sample 2
n_1 = sample size from sample 1
n_2 = sample size from sample 2
s_p^2 = pooled sample standard deviation

$(n_1 + n_2 - 2)$ degrees of freedom. The one-tailed t test would be used to determine if there is a significant difference in the mean of one group from the mean of the other.

The t test is the test method which will be used to test the hypotheses proposed in this research effort. Program cost data, obtained from program managers, was collected for each program and a mean was calculated. This mean value was than incorporated into the t statistic to test the following hypotheses:

H_0: A significant cost overrun is independent of a sound SQA program; M1 = M2.

H_a: A significant cost overrun is not independent of a sound SQA Program; M1 < M2.

H_0: A cost overrun is independent of the baseline management standard; M1 = M2.

Ha: A cost overrun can be reduced by applying baseline management standard; M1 < M2.

Likewise, information regarding schedule slips were obtained for each program and a mean calculated in the following t tests:

> H_0: A schedule slip will be independent of a sound SQA program; M3 = M4.
>
> H_a: schedule slip can be reduced by a a sound SQA program; M3 < M4.
>
> H_0: A schedule slip will be independent of the baseline management standard; M3 = M4.
>
> H_a: A schedule slip can be reduced by applying the baseline management standard; M3 < M4.

While such concepts, as implied by the words "sound" and "standard", are difficult to quantify, the design, construction and validation of the questionnaire serves as the basis by which such concepts are defined. As explained above, those concepts identified in the literature as significantly contributing to an effective SQA program and/or an effective baselining policy were incorporated into the questionnaire. Based on such criteria, each sample program was evaluated to determine the effectiveness of its SQA and baseline management policies.

The final aspect of this research methodology concerns exploration into the question:

> Can a SQA program and a baseline management program be integrated into a single plan which will result in greater benefits than the individual application of either of these methodologies?

With this concern being primarily exploratory in nature, the purpose was to develop concepts more clearly and provide greater insights into the relationships between the variables of SQA and baseline management. The methodology used was an opinion survey relating to the need for some relationship to exist between SQA and baseline management. The individual responses will be reported in chapter five.

Summary

This chapter has described the research methodology used for this research effort. The intent was to describe the thesis methodology in sufficient detail so as to permit a reader to accomplish a complete replication of this research. This chapter included a discussion of the design, construction, validation and utilization of the questionnaire within the context of a structured interview format and a discussion of the statistical test, associated assumptions and the sampling technique utilized.

This effort was proposed as a cross-sectional study of an ex post facto design. The primary assertion of this research effort is that an SQA plan together with baseline management can be integrated into a program whereby a software program's cost and schedule integrity can be maintained, the computer program's reliability enhanced, and its maintenance requirements reduced during the operational phase of its life. In the next chapter, data handling and analysis are discussed.

V. Findings and Analysis

Introduction

Analysis of the collected data is presented in this chapter. This analysis will be presented in two stages. First, a discussion of collected data will be conducted to familiarize the reader with the actual survey instrument utilized; describing the rationale used to decide if a particular program should be classified as having a sound or an unsound SQA program and as following or not following the baseline management standard, and summarizing classification findings. Second, having categorized the findings, the test statistic will be presented and a determination made to either accept or reject the hypotheses. Conclusions and recommendations derived from this analysis will be presented in the subsequent chapter.

Data Collection

The collected data consists of twenty-one interviews with Program Managers (PMs). All interviewed PMs were managing programs requiring the development of software for Embedded Computer Systems with the program having at least progressed well into Full Scale Development. The survey instrument utilized throughout the course of the interviewing process is presented in Appendix A. The questions contained within the instrument were designed to explore the key aspects of a program with respect to SQA and baseline management. Each interview is classified in terms of

TABLE VII

Program Classification Summary

Program	Sound SQA	Standard Baseline	Comments
1.	No	No	
2.	No answer	No	
3.	No answer	No	
4.	Yes	No	Not used; see program analysis in Appendix B.
5.	No	No	
6.	No	No	
7.	Yes	No	
8.	No	Yes	
9.	No	No	Not used; see program analysis in Appendix B.
10.	N/A	N/A	Not used; see program analysis in Appendix B.
11.	No	No	
12.	Yes	No	
13.	No	No	
14.	Yes	No	
15.	Yes	Yes	
16.	No	Yes	
17.	No	Yes	
18.	No	No	
19.	No	Yes	
20.	No	Yes	
21.	No	No	

both SQA and baseline management. Specifically, a program is classified as either having a sound or unsound SQA program and as following or not following the baseline management standard. The rationale for these classifications is contained in Appendix B. In addition, Appendix B also contains cost and schedule information, which is used for input into the test statistic. Table VII summarizes the SQA and baseline management classifications as contained in Appendix B.

Significant Cost Overrun and Schedule Slip

TABLE VIII

Program Managers' Opinions of What Constitutes Cost Overruns and Schedule Slips

Program	Cost Over-run	Schedule Slip	Program	Cost Over-run	Schedule Slip
1.	15%	3 mo	11.	NR	NR
2.	NR	NR	12.	30%	2 wk
3.	NR	NR	13.	20%	NQ
4.	5%	NQ	14.	15%	3 mo
5.	10%	NQ	15.	NR	NR
6.	25%	3 mo	16.	10%	NQ
7.	20%	NQ	17.	15%	4 mo
8.	NQ	NQ	18.	10%	5 mo
9.	15%	3 mo	19.	15%	4 mo
10.	2%	3 mo	20.	10%	NQ

NR: No response.
NQ: Response was not quantifiable. See Appendix B for explanation.

Only nine of the 21 interviews resulted in useable responses to find an average for significant a schedule slip. Based on the nine quantifiable responses, on the average, 3.11 months is considered a significant schedule slip. It should be noted that the indicated schedule slips, contained in Table VIII, are in relation to end item delivery dates. Also, many of the program managers caveated there estimate, saying that a schedule slip is really only crucial or significant when the user or mission is impacted. Therefore, for schedule slips, a slip beyond the needed delivery date was considered a better measure than a schedule slip as a percentage of the contract schedule period. For example, a three month slip on a 12 month program may not have the same impact that a one month slip would have on a compressed 36 month program with an urgent user requirement. Conversely, a one month slip on a critical 12 month program may cause serious repercussions while a three month slip on a less crucial, 36 month program would have no impact at all. What needs to be accounted for is the urgency of the program and the user's need for the end item.

Fifteen of the 21 program managers interviewed provided useable estimates for calculation purposes of what they considered to be a significant cost overrun. The average of these responses is 14.46 percent. Appendix B provides additional insights as to what constitutes a significant cost overrun or schedule slip.

Statistical Analysis

As presented in the chapter four of this thesis, the test statistic to be utilized in the determination of either accepting or rejecting thesis hypothesis was the one-tailed t test for small sample inferences about the difference between two population means. This statistic is of the form:

$$t = \frac{x_1 - x_2}{\sqrt{s_p^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

$\bar{x}$ =	sample mean
s =	sample standard deviation
n =	sample size
df =	degrees of freedom
t =	test statistic
s_p^2 =	pooled standard deviation for small samples

<u>SQA Cost Analysis</u>.

1. Null Hypothesis. H_0: A significant cost overrun is independent of a sound SQA program; M1 = M2.

   H_a: A signigicant cost overrun will not occur if a sound SQA program is administered; M1 > M2.

   Reject H_0 if t > the critical t value.

2. Significance level. alpha = 0.05 (one-tailed test)

3. Calculated value.

$$t = \frac{3.25 - 2.683333}{\sqrt{24.135514\left(\frac{1}{4} + \frac{1}{12}\right)}}$$

$$= \frac{.566667}{2.8364}$$

$$= .1997838 \text{ with d.f.} = 14$$

4. Critical test value. Critical t value = 1.761

TABLE IX

SQA Cost Data Summary

Sound SQA Program	Sound SQA Overrun	Unsound SQA Program	Unsound SQA Overrun
7.	\$ 0	1.	\$11
12.	0	5.	0
14.	13	6.	7.2
15.	0	8.	0
		11.	0
		13.	0
		16.	0
		17.	14
		18.	0
		19.	0
		20.	0
		21.	0
Total	\$13	Total	\$32.2

$\bar{x}_1 = 3.25$ $\bar{x}_2 = 2.683333$
$s_1 = 5.629$ $s_2 = 4.85143$
$n_1 = 4$ $n_2 = 12$

$s_p^2 = 24.135514$

5. Decision. Since the calculated value is less than the critical value (.1997838 < 1.761), the null hypothesis is accepted.

Showing that a sound SQA program could cause a reduction in significant cost overruns was the initial reason for this test. After the data was collected, however, the mean cost overrun of the programs with a sound SQA program was greater than the mean of programs not having a sound SQA program. Thus, immediately a judgement can be made that the data does not support the premise that a sound SQA program will reduce cost overruns. However, a test was carried out to determine if enough evidence was collected which would indicate that a sound SQA program would actually have an adverse effect on program costs. As the test shows, however, there is no statistical evidence that a sound SQA program has an adverse impact on program costs.

SQA Schedule Analysis.

Table X

SQA Schedule Data Summary

Sound SQA Program	Slip	Unsound SQA Program	Slip
7.	0 mo	1.	0 mo
12.	0	5.	0
14.	0	6.	0
15.	0	8.	0
		11.	0
		13.	7
		16.	12
		17.	12
		18.	0
		19.	6
		20.	7
		21.	0
Total	0 mo	Total	44 mo

$\bar{x}_1 = 0$ $\quad \bar{x}_2 = 3.6666667$

$s_1 = 0$ $\quad s_2 = 4.6607105$

$n_1 = 4$ $\quad n_2 = 12$

$s_p^2 = 17.06746$

1. Null hypothesis. H_0: A schedule slip will be independent of a sound SQA program; M3 = M4.

H_a: A schedule slip can be reduced by a sound SQA program; M3 < M4.

Reject H_0 if t < critical -t value.

2. Significance level. alpha = 0.05 (one-tailed test)

3. Calculated value.

$$t = \frac{0 - 3.6666667}{\sqrt{17.06746\left(\frac{1}{4} + \frac{1}{12}\right)}}$$

$$= \frac{-3.6666667}{2.3851946}$$

$$= -1.537261 \text{ with d.f.} = 14$$

4. Critical test value. Critical t value = -.1761

5. Decision. Since the calculated value greater than the critical value (-1.537261 > -1.761), the null hypothesis is accepted.

Based on the test result, there is <u>no statistical</u> evidence that a sound SQA program effects better schedule management.

<u>Baseline Cost Analysis</u>.

1. Null Hypothesis. H_0: A significant cost overrun is independent of the baseline management standard; M1 = M2.

H_a: A significant cost overrun will occur by applying the baseline management standard; M1 > M2.

Reject H_0 if t > critical t value.

2. Significance level. alpha = 0.05 (one-tailed test)

3. Calculated value.

$$t = \frac{2.8 - 2.6}{\sqrt{22.014556 \left(\frac{1}{5} + \frac{1}{13}\right)}}$$

TABLE XI

Baseline Cost Data Summary

Baseline Standard Program	Overrun	Not Baseline Standard Program	Overrun
15.	$ 0	1.	$11
16.	0	2.	0
17.	14	3.	0
19.	0	5.	0
20.	0	6.	7.2
		7.	0
		8.	0
		11.	0
		12.	0
		13.	0
		14.	13
		18.	0
		21.	0
Total	$14	Total	$31.2

$\overline{x}_1 = 2.8$ $\overline{x}_2 = 2.6$

$s_1 = 5.6$ $s_2 = 4.6612$

$n_1 = 5$ $n_2 = 13$

$s_p^2 = 22.014556$

$$= \frac{.2}{2.4691}$$

$= .081$ with d.f. = 16

4. Critical test value. Critical t value = 1.746

5. Decision. Since the calculated value is less than the critical value (.081 < 1.746), the

The initial intent for this test was to show that the baseline management standard would contribute to better management of program costs. However, because the mean cost overrun of those programs adhering to the baseline management standard was greater than the mean of the other programs, this could not be tested for. Instead, a test was completed to determine if the data would provide sufficient evidence that application of a baseline management standard would adversely impact program cost. However, based on the statistical test result, it must be concluded that a cost overrun is independent of the baseline management policy followed.

Baseline Schedule Analysis.

1. Null Hypothesis. H_0: A significant schedule slip is independent of applying a baseline management standard; M1 = M2.

   H_a: A significant schedule slip will result by applying the baseline management standard; M1 > M2.

   Reject if t > critical t value.

2. Significance level. alpha = 0.05 (one-tailed test)

3. Calculated value.

$$t = \frac{7.4 - .5384615}{\sqrt{7.3494661\left(\frac{1}{5} + \frac{1}{13}\right)}}$$

$$= \frac{6.8615385}{1.4406403}$$

$= 4.76283$ with d.f. = 16

4. Critical test value. Critical t value = 1.746

5. Decision. Since the calculated value is greater than the critical value ($4.76283 > 1.746$), the null hypothesis is rejected.

TABLE XII

Baseline Schedule Data Summary

Baseline Standard Program	Slip	Not Baseline Standard Program	Slip
15.	0 mo	1.	0 mo
16.	12	2.	0
17.	12	3.	0
19.	6	5.	0
20.	7	6.	0
		7.	0
		8.	0
		11.	0
		12.	0
		13.	7
		14.	0
		18.	0
		21.	0
Total	37 mo	Total	7 mo

$\bar{x}_1 = 7.4$ $\bar{x}_2 = .5384615$

$s_1 = 4.4542$ $s_2 = 1.865285$

$n_1 = 5$ $n_2 = 13$

$s_p^2 = 7.3494661$

The initial intent of this test was to provide sufficient evidence that application of the baseline management standard would curtail a significant schedule slip. Because the mean of those programs applying the baseline management standard was greater than the mean of programs not applying the baseline management standard this hypothesis was immediately abandoned. Instead, a test was conducted to determine if application of the baseline management standard will cause schedule slips. Based on the statistical test, it appears that adherence to the baseline management standard does have an adverse impact on program schedules.

Software Quality Assurance and Baseline Management Combined

With the analysis of SQA and baseline management completed, the effect of combining these disciplines will now be discussed. The hypothesis proposed implied that the combined effect of SQA and baseline management significantly impacted cost and schedule adherence during the software development process. Specifically, the hypothesis stated that, through a combination of SQA and baseline management, a synergistic effect resulted which would further ensure a program would remain within cost and on schedule. Consequently, this benefit was explored with the data gathering instrument.

While it was possible to gather cost and schedule information on programs, studying the combined relationship between SQA and baseline management proved more difficult. Consequently, this facet of the research may be classified as primarily exploratory in nature, with an objective of seeking program manager ideas on the important issues and aspects of this subject area.

Through the use of the data gathering instrument, information was obtained from program managers on the potential synergistic effect of combining SQA and baseline management. The inputs indicated overwhelmingly that a definite need exists for some form of a relationship between the groups responsible for SQA and baseline management. However, while a definite need was expressed for some type of interaction between SQA and baseline management, not a single PM could identify any specific activities or constraints which could inspire that synergistic effect and further insure that a software product would be developed within cost and on schedule. A need exists for type of interaction between SQA and baseline management, but further research is required to precisely define that interaction.

Summary

Having to reject an hypothesis is disappointing, but it is as important to show that there exists no substantial evidence to support commonly held beliefs. This does not mean that such assertions are incorrect, only that there is no data within this sample which provides statistically valid support. Originally, the research was designed to show that sound SQA and application of the baseline management standard would each contribute independently to minimizing cost overruns and reducing schedule slips. At a

95 percent confidence level, only for the effect of SQA on schedules, could a statistical test of this expected positive relationship be structured. Even then, there is no statistically significant support in favor of this positive relationship.

For the other three statistical test groups, which were considered to possess the favorable characteristic, it was immediately shown that no such positive relationship can be expected. In fact, the opposite effect is suggested. Therefore, testing for each of these, also at the 95 percent confidence level, was completed to determine if SQA has an adverse impact on cost and if baseline management has an adverse impact on cost and schedule integrity. The test results indicate that only one of these, the adherence to the baseline management standard on schedule, has a negative effect. However, this finding is questionable. As is explained in the next chapter, these schedule slips may be explained by other variables. Therefore, the general conclusion, based on the statistical analyses at a 95 percent confidence level, is that cost overruns and schedule slips will occur independently of a sound SQA program or adherence to the baseline management standard.

In summary, data analysis in this chapter found that on the average 14.46 percent over target price is considered a significant cost overrun. (An interesting statement, made by two program managers, is that there is no such thing as a cost overrun, only contract modifications.) On the average, 3 months beyond target delivery date is considered a significant schedule slip. However, almost every response was caveated, saying that a slip was really significant only when the mission was impacted. One program manager commented that, while no apparent mission impact may be realized when a slip occurs, there is the undesirable cost of maintaining the old equipment.

In this chapter, it was also shown that, in general, SQA and baseline management are believed to share a close relationship. No empirical data could be obtained to test for the effect of such a relationship. Finally, except for the suggested adverse effect of the baseline management standard on schedule, the statistical analyses done in this chapter indicates that cost overruns and schedule slips occur independent of SQA and baseline management. However, while the tested data failed to substantiate positive effects of SQA and baseline management on software cost and schedule control, this report would be incomplete without evaluating other available data to assess if other intervening variables might have biased the findings. The discussion of such analyses and findings are the next chapter's subjects.

VI. Conclusions and Recommendations

Evaluation of Research Accomplishment

The exorbitant cost of software, reported by high level Department of Defense (DoD) officials, inspired this research effort. These critics attribute the high software acquisition and maintenance costs to poor management in the development phase. The results are development cost overruns and delivery of poor quality software that is ineffective and difficult to maintain. The challenge put forth to procure affordable, reliable, and maintainable software provided the focus and direction for this project.

A review of topical literature emphasized the contributions of SQA and configuration management to effective control of software development and the extended benefits into the operations and maintenance phase. This literature also provided a framework for understanding what constitutes a sound SQA program and the development of a standard baseline management scheme. Comprehension of these two factors was important in developing the test hypotheses and the data gathering instrument.

As discussed in chapter one, the time constraint restricted the study to the effects of a Software Quality Assurance (SQA) program and the application of a standardized baseline management plan only during the development phase. Investigation for the effects of these two variables on the operations and maintenance phase are left for follow-on work. The objective of this thesis was to obtain the evidence necessary to substantiate that SQA and baseline management independently contribute to effective management and that, together, they result in even greater control over cost and schedule integrity.

The data was collected and analyzed in an unbiased manner. Deviations were highlighted by the researchers and any exceptions taken were appropriately explained. The individual test results were derived from small samples. Any generalizations made are only representative of programs managed at Aeronautical System Division and can only be related to the development phase. These results should not be used to presume any conclusions about circumstances involving cost or schedule in the operations and maintenance phase.

The researchers were able to complete their entire project. However, one shortcoming of this effort is that no empirical data could be collected to evaluate the impact of a combined SQA and baseline management program. Although responses from each interview indicated with universal agreement that such a combination would have a positive, synergistic effect, and, indeed that it does or should exist, no evidence of such a relationship being present in any of the programs was obtained. While none of the test results supported any of the expected relationships and even suggests that a negative schedule impact can result from following existing guidance, this information is useful for planning purposes.

Serendipity

> . . . Serendipity-- the exciting occurrence of finding useful or agreeable ideas or experiences that were not being sought. Often, the important advances in science and other disciplines come unexpectedly. An experiment fails, or an unusual and unexpected result is obtained, and to the alert researcher this may start a new train of thought that leads to a new and important discovery. The key here is that the researcher must retain at all times a sensitive curiosity which is stimulated to examine these unexpected results. Moreover, his interest and perspective must be broad enough to visualize the possible social or scientific significance of the unusual fact that the discovery may portend [33:47].

Software Quality Assurance. As evaluated through the course of this research, SQA for a particular program has been reviewed with regard to impact on program cost and schedule integrity. Consequently, from the statistical analysis presented in chapter five, a determination was made that, regardless of the excellence of SQA for a given program, the established cost and schedule for a program remain unaffected. However, while the quality of an SQA program did not appear to affect program cost and schedule, several key observations may be made concerning the degree of effectiveness of an SQA program.

Observations: Sound SQA Programs.

1. In terms of management support, the fact that the Program Manager (PM) had upper level AF management support did not appear significant. However, upper level management support within a contractor's organization did appear to reduce problems associated with software development and assist in producing a higher quality software product.

2. The amount and variety of experience of the PM, in terms of years of program management and software programs previously managed, appeared to influence the overall quality of the SQA program. Experience assisted the PM in the recognition of key facets of software development which contribute to both the development of a quality software product and the on-schedule and within-cost production of a software product.

3. The SQA programs classified as "Sound" all possessed what PMs described as "workable" SQA plans from which the intent of an SQA program could be developed. Further, for "Sound" SQA programs, PMs had usually investigated a contractor's SQA program, personally reviewed the contractor's proposed SQA plan and then worked together with the contractor to develop what a PM would consider a "realistic" SQA plan.

4. When program requirements and funding permitted the AF to contract an IV&V team, the contributions of this team generally contributed significantly to the development of a higher quality software product and to a software product being produced both on schedule and within cost.

5. The contractor's software quality assurance section also made significant contributions. An independent SQA section within a contractor's organization, especially when supported by upper level management, appeared a key factor in positively contributing to software development efforts through the early detection and correction of errors.

6. For software development efforts classified as "Sound", a contractor policy focusing on the early detection and correction of software deficiencies appeared significant. The primary reason given by PMs for contractor implementation of such a policy was that contractor upper level management appeared to recognize potential cost savings derived from the early detection and correction of software discrepancies.

7. For all PMs interviewed, all formal and informal reviews and audits were generally conducted in a satisfactory manner. Further, in several instances where a contractor was not prepared for a review and the review was not accomplished to the satisfaction of the PM, a PM would not permit software development progression until a rescheduled review was satisfactorily accomplished.

Observations: Unsound SQA Programs.

1. Programs in the "Unsound" category generally had PMs who were inexperienced with software development efforts. Consequently, a recognition of the potential of SQA did not exist, and a lack of emphasis concerning SQA was evident. Further, such PMs were often unaware of even the most general contents of their program's SQA plan, to the extent that some PMs were not even aware of the existence of an SQA plan for their program.

2. SQA plans for "Unsound" programs were generally viewed by PMs as "show" items and not supported by contractor upper level management. In some cases, an SQA

plan did not exist at all. For existing SQA plans in this category, PMs often stated that the SQA plan on their program was only given a cursory review.

3. On several programs, the interaction between the contractor's SQA activity and the contractor's engineering activity was of concern. Specifically, some contractor's had newly-formed SQA functions which handled responsibilities previously handled by the engineering function. Some hostility existed between these two sections which resulted in a lack of cooperation and detracted from the overall software development effort.

4. For "Unsound" SQA programs, the contractor's attitude was generally one that did not emphasize the early detection of software deficiencies. Reasons provided by PMs for this attitude included existing budget and schedule pressures and a lack of upper level management concern by the contractor.

Observations: General.

1. The most significant general observation noted by these researchers concerned the definition of Software Quality Assurance as interpreted by PM's. With regard to hardware, the literature reviewed was precise in defining quality assurance and how it is derived. However, when it comes to software, confusion appears to exist at the PM level as to what is SQA and how it is obtained or accomplished.

2. The degree of software development experience possessed by a PM appeared to influence the quality of a soft-

ware product and program cost and schedule considerations. Specifically, with increased experience comes an increase in emphasis on SQA and an increase in recognition of those facets which contribute to the production of a quality software product.

3. According to the literature, a reader concerned with the topic of AF SQA is left with the impression that MIL-S-52779A is a significant part of the answer in solving software development problems in the AF. However, even in those SQA programs considered "Sound", MIL-S-52779A was not always used. Nor did the SQA plans for "Sound" software development programs always contain all the elements presented for review in MIL-S-52779A.

4. As presented in the literature, Software Metrics is considered by the scientific community as in the beginning stages of development. This conclusion was evident throughout the course of all the interviews conducted. Specifically, on only three of the 19 programs reviewed were Software Metrics used and then, according to the respective PMs, Metrics were used only in a limited fashion.

5. The type of contract let for the development of a software product has a definite effect on the resultant quality associated with that developed product. Specifically, a firmed fixed price contract appeared to result in higher quality of the software product, according to several of the PMs interviewed.

6. In two programs, programs one and seventeen, to

assist in the development of quality software, PMs withheld progress payments when software development was not progressing in a satisfactory manner. The withholding of progress payments appeared to positively contribute to the overall quality of the software product.

<u>SQA Discussion</u>. As previously mentioned, this thesis has investigated the impact of the degree of excellence of an SQA program on the cost and schedule integrity of a given software development effort. And, while the analysis in chapter five suggested no significant relationship between the degree of excellence of an SQA program and the cost and schedule integrity of a program, what is disturbing is the large number of programs classified as having "unsound" SQA programs. Consequently, although many factors may account for this rating, it appears that a problem does currently exist in the area of SQA. To evaluate this problem, a more global assessment of this situation is needed.

In the SQA literature reviewed, the suggestion was made that quality software was obtained through an objective, measurable assessment of a software product and by following managerial procedures through the course of the software development process. DoD efforts to build quality software presently focus on following managerial procedures. Therefore, DoD is focusing on only half of the problem-- the procedural development of software. What is also needed is to objectively define software quality attributes.

Specifically, DoD needs to formally define software quality as a total, tangible concept. Currently, in all facets of software development within the program office, real confusion exists with regard to the precise meaning of SQA. Lack of clear definition of SQA and an apparent lack of trained and experienced software development personnel seem to be the key contributing factors to the continued confusion associated with software development.

<u>Baseline Management Discussion</u>. Like SQA, the primary research objective concerning baseline management was to evaluate its effect on cost and schedule integrity. The statistical inferences were discussed in chapter five; however, other interesting information and observations were uncovered in the course of interviewing and are shared here.

1. When queried as to what guidance was used in establishing their baseline management plans, the majority of program managers stated that MIL-STD-483 and MIL-STD-490 were used. Neither AFR 65-3 or AFR 800-14 were indicated as the reference sources in any of the interviews.

The accumulated data does not support the program managers' claim that baseline actions are in accordance with Military Standard 483. Further, Military Standard 490 does not provide guidance for baseline control. Therefore, this evidence supports the observation, reported earlier, that existing documents are not clear in their guidance or deviations are being taken by program managers. Additionally, it is important to note, that it is not recognized, by program managers, that AFR 800-14 provides the guidance which should be adhered to.

2. Without prompting, some program managers explained why the allocated baseline was delayed beyond PDR, and, in some cases, as late as PCA/FCA . According to these managers, control of an allocated baseline earlier than FCA/PCA is too early and will certainly result in costly engineering change proposals. It was confirmed that development specifications (B_5) were reviewed and monitored, but the contractor had the latitude to change the requirements in the specification as needed.

Of the 13 programs which deviated from the baseline management standard, only two had a cost overruns. Thus, it seems that the claim may be legitimate; however, it was not determined if the delivered product met original performance requirements, or, because of the limited control which the government had, if the delivered product was less than satisfactory. If this information had been solicited, a determination could have been made as to the effectiveness of this approach. That is, while a cost overrun is avoided, does the program result in a useable product which fully satisfied performance requirements. If not, would an ECP that is initiated at the end of development be more costly than would an ECP initiated earlier in the program, and would any schedule slip incurred now have been avoided by initiating an ECP earlier?

3. Surprisingly, analysis shows the effect of following the standard baseline guidance, to have a negative consequence. The data finds four of the five programs which followed the standard to have schedule slips. In reviewing other program data, however, two alternative explanations are offered for this unexpected finding.

First, two of the four programs had established baselines more rigidly than the standard guidance suggests. Thus, while following the standard may be beneficial, too rigid an application may have an adverse impact, as the data suggests.

A second, more plausible explanation, however, may be that a realistic schedule is a significant moderating variable. This observation is based on the fact that three of these four programs reported not having a realistic schedule going into the program. Credence for this explanation is also provided by program 13. It is the only program categorized as not following the standard which has a schedule slip, but it also reports not having a realistic schedule going into the program.

Other. Other unexpected discoveries were made in the course of investigation. While not specifically related to the effect that SQA or baseline management have on cost and schedule integrity, these findings suggest provocative, alternate reasons which may account for program perturbations not considered by the researchers, and highlight interesting, noteworthy observations.

1. The collected data disproves what the literature suggests. No overwhelming evidence was collected which substantiates the criticism of software as being the 'long pole' or as a major expense item for the government. Consideration of the following, however, must be made. First, the data collected only accounts for the development phase of software. Therefore, while it appears that control of cost and schedule during development is being achieved, study into the operation and maintenance of software is needed before any real conclusion can be drawn. Second, the programs were examined only by cost to the government. In many cases, the program managers stated that the contractors were experiencing cost overruns not chargeable to the government. Thus, a study of industry may disclose evidence which supports observations that software is a management problem.

2. If a program does not have realistic budget, schedule, or performance specification, then, no matter how good an SQA program or strong the baseline control, a cost overrun and schedule sl'p would be inevitable. The collected data does provide good evidence that this premise is true for schedule slips but not for cost overruns. Of the six programs used in statistical analysis and reported to have a schedule slip, five were also identified as not having a realistic ingoing program schedule. Of the four programs used in statistical analysis and reported to have a cost overrun, only one reported having an unrealistic budget

going into the program. However, review of the explanatory notes (in Appendix B of chapter five) for programs 4 and 9 indicate that the original programs were not properly structured and after renegotiation program 4 doubled in price and program 9 received one tenth of the originally required products. It is difficult to determine a reasonable estimate of what the overrun was for these programs, but it is obvious that substantially more was paid for the products finally received than was originally planned to be spent.

Directions for Future Research

Through the course of this research, it became apparent that the area of SQA and baseline management, as related to software development, contained many possibilities for future research. In conclusion, the following suggestions are presented as topics for future research:

1. Clearly, the next step is to study what happens to software during the operation and maintenance phase. If the performance of the developed software examined in this thesis could be evaluated out in the field, the results could be related to the development characteristics recorded here and long term effects could be analyzed. Such an analysis would provide important insight into the relationship between early program control and decisions and out year effects in terms of life cycle cost effectiveness. How well did the software meet performance requirements and how maintainable was the software?

2. While this thesis effort reviewed and presented the

idea that SQA was a two component function of objectively assessing software quality and procedurally managing software quality, empirically this concept was only partially explored. Consequently, it is suggested:

a. that a literature review and progress analysis be undertaken to objectively assess past and present DoD efforts to incorporate and utilize software metrics.

b. that programs utilizing software metrics be analyzed in terms of the extent of metric usage and the extent of user satisfaction.

c. that an attempt be made to precisely define what is meant and what should be meant by the term "software quality" within the DoD.

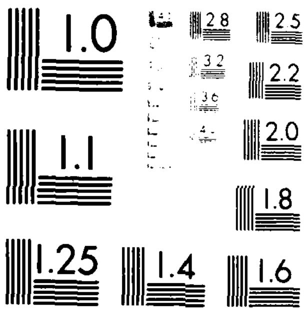
3. The concept of quality, as presently interpreted by the DoD, needs to be researched in terms of user satisfaction. Consequently, analysis related to the degree of excellence of an SQA program during software development, training and experience of PMs when managing software development efforts, and the contract type should be conducted to determine their effect on user satisfaction.

4. In the course of investigation, the researchers were provided an incredible example of how market structure may affect contractor performance. From the onset of this example program, the contractor failed to meet schedule deadlines. Although a strongly worded letter of admonishment, by the ASD commander, was sent to the president of the company, the contractor's performance has never really improved. The program manager stated that the contractor is the only source capable of working the technology peculiar to contract and for this reason is not intimidated or

AD-A147 051 THE EFFECTS OF SOFTWARE QUALITY CONTROL AND BASELINE MANAGEMENT ON THE AC..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. 2/2

UNCLASSIFIED S C KIMHAN ET AL. SEP 84 F/G 5/1 NL

END
DATE FILMED
11 84
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.1
1.8
1.25
1.4
1.6

concerned by customer expressed dissatisfaction. This is an extreme example, but it does provide evidence that monopolistic power can prove detrimental to a program's management. Further compounding this problem is the decreasing number of vendors available and/or willing to work with the government. How prevalent these problems are, and their affect on cost and schedule are interesting and important subjects for future study.

5. The research was limited to analysis of programs managed at ASD. Study is required of the other Air Force Systems Command Product Divisions. Repetition of this project would contribute towards its validity. Additionally, it may uncover other important findings not revealed here.

6. Unanimously, PMs agree on the need for a relationship or working interaction between the disciplines of SQA and baseline management. However, when PMs were asked to indicate the context of the relationship and potential degree of interaction, PMs could not identify a format for establishing such a working interaction. Consequently, a need exists for further exploration to be conducted on the topic of establishing a format or context for the interaction of SQA and baseline management. This would involve defining the current role of SQA as it is both perceived and formally specified and defining the role of baseline management as it is both perceived and formally specified, and then exploring the potential of combining these two areas and the resultant effects on the cost, schedule and quality of a product.

7. Lastly, a similar research project should examine the contractor side of the software development business. As pointed out, the collected data did not support claims that software is a cost and schedule headache. Because of the way the government accounts for performance, especially in a firm fixed price arrangement, the contractor's cost overruns are not considered by the government for reporting purposes. Even for cost type arrangements, the government shares in the cost overrun only up to the ceiling, price beyond which the government is not concerned, for reporting purposes, about further overruns incurred by the contractor. In industry, however, these overruns are recognized; therefore, their historical cost data might provide evidence more in line with the concerns expressed in current literature.

Summary

In this chapter, the overall research effort was reviewed and general observations were presented as derived from the research effort. While it could not be statistically concluded that a "Sound" SQA program or following the baseline management standard influences the cost effective development of quality software, it is hoped that the observations recorded will provide useful management insights and stimulate future research. Developing quality software, which is both within cost and on schedule, remains a significant concern within the DoD. Therefore, continued research is urgently needed to address this problem.

Appendix A: Data Gathering Instrument

SECTION 1: BACKGROUND INFORMATION

1. What type of contract strategy is your program employing?

2. What price was the contract awarded for?

    2a. Of this price, how much is for software effort?

3. Would you please tell me what the initial PMRT and IOC dates were at contract award?

    3a. Have there been any slips to these dates?

4. Has the program incurred any cost overruns due to software problems?

5. Has the program experienced any schedule slips due to software problems?

6. As a program manager, what do you consider a significant cost overrun?

7. As a program manager, what do you consider a significant schedule slip?

SECTION II: SOFTWARE QUALITY ASSURANCE

Management Support.

8. Does this Program Office have a written SQA policy, which was either developed by you, the PM, or some higher level, emphasizing the overall management objectives of a SQA Program?

9. Does the contractor on this program have a written SQA policy emphasizing the overall management objectives of a SQA Program?

10. If such a contractor SQA policy existed, was it reviewed by Program Office personnel?

11. For this program, did the contractor have a specific SQA plan?

12. If such a plan existed, were standards established for documents, test coverages, and configuration management reports in the plan?

13. If such a plan existed, were test tools and test support software outlined in the plan?

14. If such a plan existed, was a sequence of formal and possibly informal reviews outlined in the plan which would delineate the progression of software development?

15. Does the contract for this program specify the use of MIL-S-52779A for software quality assurance requirements?

Independent Support.

16. On this program was an Independent Verification and Validation (IV&V) team utilized?

17. On this program did the contractor have an internal quality control section for this development effort?

18. If such a section exists, has the contractor documented a management plan to periodically assess the effectiveness of this section?

19. Does the contractor have established a software discrepancy reporting system to assure that all known problems are documented and tracked for corrective action?

Testing.

20. Does the contractor's attitude seem to emphasize the early detection of deficiencies in the software development process?

21. Does the SQA plan identify the contractor's software test activities, and if so, does the SQA plan allow these activities to be monitored for the certification that test results are the actual findings of the test activities?

22. Do you recall the test tools used in software testing?

23. Were Software Metrics used in software testing?

Software Development Progression.

24. In analyzing the developing software product throughout the software development process, was software documentation developed and maintained to your satisfaction?

25. Were formal and possibly informal reviews conducted to monitor the software development process?

26. Was a successful Preliminary Design Review conducted for each Configuration Item before proceeding into detailed design?

27. Was a successful Critical Design Review conducted for each Configuration Item prior to coding?

28. Before a new phase of software development was undertaken, was work in the preceding phases of software development accomplished to your satisfaction?

SECTION III: BASELINE MANAGEMENT

Program Baseline Management Strategy.

29. Was there a baseline management policy established prior to contract award?

30. Was this policy adhered to?

31. If yes, is there a document you can furnish us that discusses this policy?

31a. Was this policy based on any particular MIL-STD or AFR?

Program Baseline Management Actions.

32. When was functional baseline established?

33. When was allocated baseline established?

34. At what point did the contractor establish an internal product baseline?

SECTION IV: OTHER

35. Do you feel that the ingoing program had a realistic budget, schedule, and system specification?

36. Based on your experience, do you see a relationship (or the need for a defined relationship) between SQA and baseline management? If so, could you elaborate please?

Appendix B: Program Data Summary

Program 1

Contract Type:	CPIF	Program Cost:	$106M
Software Cost:	$35M	Cost Overrun:	$4M
Schedule Slip:	0 months		

Significant Cost Overrun: 15%
Significant Schedule Slip: 3 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: The PM stated that upper level management support existed behind both AF and contractor efforts on this program. However, it is questionable as to the effectiveness of this emphasis. The PM indicated that while an SQA plan existed, the plan was more of a show item. Further, it was discovered that only a cursory review of the SQA plan was conducted. Lastly, the PM indicated that while the standard formal and informal reviews and audits were conducted, in the opinion of the PM, these reviews and audits were not wholly effective due to a "build then fix" attitude of the contractor.

B. Independent support: A contracted Independent Verification and Validation (IV&V) team was used on this program, and in the opinion of the PM was effective in the identification of software problems. The contractor did have an independent SQA function, but the effectiveness of this section could not be determined.

C. Testing: Extensive testing was conducted by the AF and as a result of this testing numerous software errors were detected. In the opinion of the PM, the contractor developing this software product did not possess an attitude relating to the early identification of errors. Consequently, since testing was conducted in the latter stages of software development, detected software errors were more costly to correct.

D. Progression: As stated in the Management support section of this analysis, according to the PM, the progression of the software development process was unsatisfactory.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Used.

3. Although this program is classified as possessing an unsound SQA program, the PM stated that a quality software product was being produced. This appears to be the result of the PM withholding payments through the course of the software development effort until the software produced actually performed as required. It should be noted that as this practice was utilized by the PM, in the opinion of the PM, contractor upper level management support became more evident as evidenced by an apparent shift in the contractor's attitude toward the early detection of errors and a smoother software development progression.

<u>MEETS BASELINE MANAGEMENT STANDARD: NO</u>.

Allocated baseline was delayed till past CDR; thus, the program fails to meet the criteria of the baseline management standard.

Program 2

Contract Type:	FFP	Program Cost:	$4.3M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: No Response
Significant Schedule Slip: No Response

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: NONE.

Neither the PM nor the lead software engineer on the program choose to respond to the SQA portion of the interview. This program was not included in chapter five SQA calculations.

MEETS BASELINE MANAGEMENT STANDARD: NO.

According to the program manager, because this is an FFP contract, it is not to the program's benefit to establish baselines. The contractor should be left to manage his own baseline. The program office will establish baseline control at PCA. Thus, this program is not considered to meet the requirements of the baseline management standard.

Program 3

Contract Type:	FFP	Program Cost:	$4.9M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: No Response
Significant Schedule Slip: No Response

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: NONE.

Neither the PM nor the lead software engineer on the program choose to respond to the SQA portion of the interview. This program was not included in chapter five SQA calculations.

MEETS BASELINE MANAGEMENT STANDARD: NO.

Allocated baseline is being delayed till FCA/PCA; thus, this program does not meet the criteria of the baseline management standard.

Program 4

Contract Type:	FFP	Program Cost:	$200M
Software Cost:	Data Unavailable	Cost Overrun:	See Note
Schedule Slip:	0 months		

Significant Cost Overrun: 5%
Significant Schedule Slip: When the mission cannot be met.

Realistic Budget: No
Realistic Schedule: No
Realistic System Specification: Yes

Note: This program was originally structured as a $91M FPI contract. One year later, with congressional approval, the program was renegotiated as an FFP contract for $200M. While the program doubled in price the scope of effort remained virtually unchanged. Because of the significant change in price this program is considered an exceptional deviation and disregarded for statistical testing. However, it does provide a good example of a program with an initially unrealistic schedule and budget.

SQA CLASSIFICATION: SOUND.

A. Management support: Both the AF and the contractor appeared to have upper level management support emphasizing the significance of SQA. An SQA plan was in existence and, in the opinion of the PM, was thoroughly reviewed in a joint effort by both program office and contractor personnel. The PM indicated that, through the course of the program, the SQA plan was constantly under revision pursuant to the intent of a workable SQA program. The SQA plan included such items as the establishment of documentation standards, outlining of needed software testing tools and support software, and the methodology of the software development progression.

B. Independent support: Initially, neither a contracted nor a Program Office IV&V team was utilized; however, after the PM recognized a lack of experienced AF personnel to read, test, detect and interpret software problems within the Program Office, a contracted IV&V team was employed. The PM held that the IV&V team utilized significantly contributed to the success of the software product. Also, the contractor did have an internal SQA section, which the PM classified as effective. The contractor's SQA section had a direct link with the company president as well as having such items as a software discrepancy reporting system to document and track software problems for corrective action. In addition, the contractor had an SQA section inspection program to periodically assess the effectiveness of the SQA section.

C. Testing: All software was tested by the contractor at least twice before testing was actually monitored by AF personnel. Further, in the opinion of the PM the contractors attitude seemed to be one of emphasizing the early detection of errors.

D. Progression: Software development followed an orderly progression, which was to the satisfaction of the PM.

E. Other:

1. Software Metrics: Used.

2. MIL-S-52779A: Used.

3. This program seemed to have one of the best SQA programs with two items standing out as significantly contributing to the success of this program:

   a. The PM's recognition of the significance of SQA and determination to implement an SQA program.

   b. The PM was able to develop a sound working relationship with the contractor.

   c. The contract type appeared to force the contractor to work within cost and schedule requirements.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The allocated baseline is being delayed till FCA; therefore, it does not meet the baseline management standard's requirements.

Program 5

Contract Type:	FFP	Program Cost:	$16.2M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: Not until someone else is being seriously impacted.

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: For this program, it appeared that there was not any AF upper level management support and even the PM did not appear to emphasize the significance of SQA. The PM was not aware of any upper level management emphasis concerning SQA by the contractor. An SQA plan was not specifically developed for this program, but instead developed within the context of the Configuration Management plan.

B. Independent support: Neither a contracted nor a Program Office IV&V team was used, but the contractor did have an independent SQA section. The PM did not make a determination as to the effectiveness of the contractor's SQA section, although the PM did state that a software discrepancy reporting system existed and appeared effective.

C. Testing: All developed software was tested several times before presentation to AF personnel for testing. The PM stated that the contractor emphasized the early detection of software errors during development.

D. Progression: All formal and informal reviews and audits followed an orderly progression, which was to the satisfaction of the PM.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. This particular PM did not view SQA as overly significant. Instead, the PM emphasized the firm fixed price type of contract as the driving factor in the development of a quality software product.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The allocated baseline was established after CDR; thus, it does not meet the criteria of the baseline management standard.

Program 6

Contract Type:	CPIF	Program Cost:	$100M
Software Cost:	$20M	Cost Overrun:	$1.44M
Schedule Slip:	0 months		

Significant Cost Overrun: 25%
Significant Schedule Slip: 3 months after IOC.

Realistic Budget: Yes
Realistic Schedule: No
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: Within the AF, upper level management support did not exist. However, the contractor did have an upper level management policy emphasizing the significance of SQA. Further, the PM believed the intent of this policy guidance was adhered to by the contractor throughout the software development process, with the exception of the contractor's attitude toward early detection of errors. An SQA plan was developed for this program; however, it was not reviewed by Program Office personnel.

B. Independent support: Initially, neither a contracted nor a Program Office IV&V team was used, but, after the PM recognized the lack of expertise in the area of software development within the Program Office, a contracted IV&V team was utilized. The contractor did have an independent SQA section which he believed was effectively contributing to the software development effort. Quality documentation accompanying the software developed substantiated the PM's claim.

C. Testing: Throughout the testing process, the attitude of the contractor did not seem to emphasize the early detection of errors.

D. Progression: In the opinion of the PM, before a new phase of software development was undertaken, work in the preceding phases had not been satisfactorily accomplished. Software development did not follow an orderly progression.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. The PM did not appear knowledgeable on the topic of SQA or even of the arguments either for or against the significance of SQA. And, while the Program Office lead engineer was more knowledgeable of the SQA subject area, the complaint was that, due to a lack of time and manpower, it was exceptionally difficult to be thoroughly concerned with SQA for this particular program.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The allocated baseline was delayed beyond PDR; therefore, it does meet the criteria of the baseline management standard.

Program 7

Contract Type:	FPIF	Program Cost:	$29M
Software Cost:	$14.5M	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 20%
Significant Schedule Slip: 20% of contract period beyond IOC.

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: SOUND.

A. Management support: With the exception of the PM, AF upper level management support did not apparently exist on this program. The PM stated he was aware of upper level management support by the contractor. An SQA plan did exist for this software development effort, and the SQA plan was reviewed by Program Office personnel. The SQA plan did establish documentation standards, outline test tools and test support software and delineate the software development progression.

B. Independent support: A contracted IV&V team was used, and the contractor did have an independent SQA section. Further, the contractor did utilize, in the opinion of the PM, an effective software discrepancy reporting system to document and track corrective actions on identified software discrepancies. The contractor also had a program to periodically assess the effectiveness of the SQA section.

C. Testing: The contractor tested the software several times prior to AF monitoring of tests. The contractor's attitude seemed to emphasize the early detection of errors.

D. Progression: The PM was satisfied that, before a new phase of software development was undertaken, work in preceding phases was accomplished to his satisfaction. The PM stated that all formal and informal reviews conducted were successful and software development did follow an orderly progression.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. Although SQA practices appeared to contribute to the development of a quality software product, several other factors seemed to contribute to the success of this program.

    a. The PM appeared a hardworking, highly motivated individual who recognized the potential of SQA.

    b. The PM insisted on a realistic budget and schedule.

    c. The type of contract appeared to force the contractor to produce a quality software product.

<u>MEETS BASELINE MANAGEMENT STANDARD: NO</u>.

The allocated baseline was established at CDR; therefore, the program does not meet the criteria of the baseline management standard.

Program 8

Contract Type:	FFP	Program Cost:	$14M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 20K for this program
Significant Schedule Slip: 1 month beyond delivery date

Realistic Budget: No
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: The PM stated that there was a growing concern on the part of upper management in both the AF and by the contractor relating to SQA. The PM stated that, as a result of this concern, his awareness concerning SQA had increased. An SQA plan did exist and was reviewed by Program Office personnel. However, the PM was unaware of the contents of the plan.

B. Independent support: Neither a contracted nor a Program Office IV&V team was used on this program. The contractor did have an independent SQA section, but the PM could not comment on the effectiveness of this section.

C. Testing: Initially, the contractor appeared to have a "build then fix" attitude concerning software development. However, according to the PM, as a result of the increased concern of AF upper level management, the contractor's attitude seemed to shift to one of detecting errors as soon as possible in the software development effort.

D. Progression: After the expressed increase in concern for SQA by both AF and contractor upper management, all formal and informal reviews and audits followed a more orderly progression, which was to the satisfaction of the PM.

E. Other:

1. Software Metrics: Not used.
2. MIL-S-52779A: Used.
3. While the overall quality of the software product being produced appeared to be improving, two factors other than SQA seemed to be the driving forces.
    a. The type of contract appeared to forcing the contractor to produce a higher quality software product.
    b. Upper level management support from both the AF and contractor appeared to be contributing to the development of a higher quality software product.

<u>MEETS BASELINE MANAGEMENT STANDARD: YES</u>.

The functional baseline was established at contract award, allocated baseline at PDR and product baseline at PCA, the program therefore, adhered to the baseline management standard.

<u>Program 9</u>

Contract Type:	CPIF	Program Cost:	$21M
Software Cost:	$14.5M	Cost Overrun:	See Note *
Schedule Slip:	0 months		

Significant Cost Overrun: 15%
Significant Schedule Slip: 3 months beyond delivery date

Realistic Budget: No
Realistic Schedule: No
Realistic System Specification: No

*NOTE: Because of some significant program peculiarities, this program is disregarded for statistical testing. This program involved new technology requiring the development of sophisticated software. The original $21M program was let to procure ten systems. The risk was not properly assessed and two years after the initial contract was awarded, the effort was significantly reduced, and only one system was to be delivered with the contractor incurring any costs beyond the $21M. Additionally, no reprocurement data was procured, as new policy requires contract support be obtained for the type of software being acquired.

<u>SQA CLASSIFICATION: UNSOUND</u>.

A. Management support: AF upper level management support did not overtly exist. With regard to the contractor, initially upper level management support also did not exist. However, due to the large dollar amounts and the type of contract covering this program as combined with continued problems experienced early in the program, the contractor changed management personnel. Consequently, with this personnel change, upper level contractor management support of SQA practices improved. It should also be noted that an SQA plan did exist for this effort, but the PM was unaware of its contents.

B. Independent support: Neither a contracted nor a Program Office IV&V team was used on this program. However, in the opinion of the PM, the contractor did have an effective independent SQA section.

C. Testing: After the change in management personnel, testing practices were revised so software was tested several times prior to the monitoring of testing activities by the AF. Further, the contractor's attitude seemed to shift to the early detection of software discrepancies.

D. Progression: After the change in management personnel, software development followed a more orderly progression than had previously been experienced. Specifically, before a new phase of software development was undertaken, work in the preceding phases was accomplished to the satisfaction of the PM.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Used.

3. A comprehensive SQA program did not appear to exist on this program. Instead ,the success of this program seemed to be driven by other factors.

   a. The type of contract covering this program appeared to help the contractor realize that it was in his own best interests to utilize the potential of SQA to assist in reducing software development costs and ultimately produce a higher quality software product.

   b. The change in contractor management, which resulted in increased upper level management support by the contractor, also appeared to contribute to the production of a higher quality software product.

MEETS BASELINE MANAGEMENT STANDARD: NOT APPLICABLE.

No specifications were being procured.

Program 10

Contract Type:	FFP	Program Cost:	$31M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 2%
Significant Schedule Slip: It depends on the program.

Realistic Budget: Yes
Realistic Schedule: Optimistic schedule (not because of software)
Realistic System Specification: Yes

Note: This program required the procurement of test programs. The strategy used was to procure factory test programs. Thus, no software development was managed under this contract. Because of this peculiarity, this program is disregarded for statistical analyses.

SQA CLASSIFICATION: NOT APPLICABLE.

MEETS BASELINE MANAGEMENT STANDARD: NOT APPLICABLE.

Program 11

Contract Type:	FFP	Program Cost:	$153M
Software Cost:	$10M	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: No response
Significant Schedule Slip: No response

Realistic Budget: No
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: With the exception of the PM, upper level management support from the AF did not exist. The contractor did have upper level management policies emphasizing the significance of effective SQA practices. However, through the course of the interview with the PM, it was difficult to determine the effectiveness of upper level contractor management policies towards the production of a quality software product. Nevertheless, an SQA plan was developed and tailored to this development effort, and the SQA plan was reviewed by Program Office personnel. The plan did include such items as the establishment of standards for documentation, the tools and support software to be used in testing developed software, and the sequence of reviews and audits to be conducted.

B. Independent support: A contracted IV&V team was used on this software development effort, and in the opinion of the PM was effective. The contractor also had an independent SQA section; however, the PM could not comment on the effectiveness of this section.

C. Testing: All tests were not accomplished to the satisfaction of the PM. The PM noted significant problems were experienced in the area of integration of the software and hardware. While the contractors attitude appeared to emphasize the early detection of software discrepancies, this software/ hardware integration problem appeared to detract from the overall success of this program.

D. Progression: The PM felt that the software developed for this program was taken too lightly by the contractor. Consequently, the PM stated that although the provisions of the contract were being fulfilled, software development was not following an orderly progression and work was not being satisfactorily accomplished.

E. Other:

1. Software Metrics: Not used.
2. MIL-S-52779A: Not used.
3. The type of contract appeared to have more of an influence on the development of the software associated with this program than the attempts at implementing an SQA program.

MEETS BASELINE MANAGEMENT STANDARD: NO.

Establishment of the allocated baseline was delayed till FCA/PCA. Thus, the program fails to meet the requirements of baseline management standard.

Program 12

Contract Type:	FPIF	Program Cost:	$39M
Software Cost:	$19.5M	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 30%
Significant Schedule Slip: 2 weeks

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: SOUND.

A. Management support: From an AF perspective, upper level management support did not exist concerning SQA. However, the contractor on this program did have a written SQA policy which, according to the PM, did highlight the significance of developing quality software. An SQA plan was developed specifically for this program and was reviewed in detail by the PM. The plan did contain such items as documentation standards, tools and support software needed for testing and the formal and informal reviews to be conducted through the course of the software development.

B. Independent support: A contracted IV&V team was used and was effective. Also, the contractor did have a section specifically dedicated to SQA, which the PM also classified as effective.

C. Testing: All software developed was tested several times prior to being tested with AF review. The PM believed that the contractor strongly emphasized the early detection and correction of errors. The PM felt that the contractor had recognized that it was in the best interest of the company to correct errors early.

D. Progression: All formal and informal reviews and audits were conducted to the satisfaction of the PM. Software development followed an orderly progression. The PM stated that before a new phase of software development was undertaken all work in preceding phases was accomplished in a satisfactory manner.

E. Other:
1. Software Metrics: Metrics were used by the IV&V contractor.

2. MIL-S-52779A: Not used.

3. Although upper level AF management support did not overtly exist for this program, several other factors strongly contributed to this program.

a. The type of contract seemed to help insure that a quality software product was being produced.

b. Before this program started, the PM insisted on a realistic budget and schedule.

c. The PM had nineteen years experience in the area of program management and appeared to recognize the problems typically associated with software development.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The establishment of allocated baseline was at FCA/PCA. Thus, the program fails to meet the criteria of the baseline management standard.

Program 13

Contract Type:	FFP	Program Cost:	$2M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	7 months		

Significant Cost Overrun: 20%
Significant Schedule Slip: 25% of contract schedule beyond delivery date

Realistic Budget: Yes
Realistic Schedule: No
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: AF upper level management support did not exist to assist the PM. The contractor did at least verbally espouse the significance of SQA. However, according to the PM, contractor emphasis was ineffective in implementing any form of an SQA program. The PM noted that a significant problem existed within the contractors plant; although an independent SQA function existed, this function was newly created and in strong conflict with the contractors engineering function which previously had been responsible for quality assurance efforts. It should be noted that an SQA plan did not exist for this program and that the documentation produced to accompany the developed software was generally unsatisfactory.

B. Independent support: Neither a contracted nor Program Office IV&V team was used. And, although the contractor did have an independent SQA section, for reasons previously mentioned, this section was classified by the PM as ineffective. In fact, the PM believed the existence of the contractors SQA section and the constant conflict with the contractors engineering section actually detracted from the program.

C. Testing: The contractor's attitude was one of "build then fix" the software.

D. Progression: Although formal reviews were accepted as complying with contractual requirements, the quality of the software product being produced was not to the satisfaction of the PM. According to the PM, the contractor tried to do many things at the same time and consequently hardware integration problems resulted. The PM concluded that software development efforts did not follow an orderly progression.

E. Other:
1. Software Metrics: Not used.
2. MIL-S-52779A: Not used.

3. The PM did not seem all that knowledgeable on the topic of software development. Further, the type of contract seemed to drive any quality inherent in the software product.

<u>MEETS BASELINE MANAGEMENT STANDARD: NO</u>.

The allocated baseline was established at PCA. Thus, the program does not meet the requirements of the baseline management standard.

<u>Program 14</u>

Contract Type:	FPI	Program Cost:	$36M
Software Cost:	$3M	Cost Overrun:	$400K
Schedule Slip:	0 months		

Significant Cost Overrun: 15%
Significant Schedule Slip: 3 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

<u>SQA CLASSIFICATION: SOUND</u>.

A. Management support: Upper level management support existed for both the AF and contractor. For the AF, verbal emphasis existed. The contractor on this program actually had a written SQA policy which effectively conveyed the significance of developing quality software. This program did have an SQA plan which was reviewed by Program Office personnel and deemed satisfactory.

B. Independent support: According to the PM, a contracted IV&V team was used effectively on this effort. The contractor did have an independent SQA section, which the PM believed was effectively contributing to software development.

C. Testing: All software was tested several times prior to AF review. The PM stated that the attitude of the contractor was focused on the early detection of errors.

D. Progression: All formal and informal reviews and audits were conducted to the satisfaction of the PM. In the opinion of the PM, all phases of the software development effort were completed to a satisfactory level before a new phase of software development was undertaken. Software development followed an orderly progression.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. The PM was a highly experienced individual with more than twenty years in the area of program management. This individual appeared to recognize the potential problems associated with software development. The type of contract also appeared to drive the quality of the product.

<u>MEETS BASELINE MANAGEMENT STANDARD: NO</u>.

The allocated baseline was established at CDR; therefore, the program does not meet the criteria of the baseline management standard.

<u>Program 15</u>

Contract Type:	FPI	Program Cost:	$13M
Software Cost:	$13M	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: No response
Significant Schedule Slip: No response

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: SOUND.

A. Management support: With the exception of the PM, AF upper level management support did not exist for this program. However, the lack of overt upper level AF support did not hinder this program. The PM had extensive training in the area of software development along with more than five years on the program. Consequently, it appeared that the PM had a complete understanding of the potential problems associated with software development as well as an understanding of how to avoid such problems. The contractor did have a written SQA policy, which in the opinion of the PM, effectively conveyed the significance of building quality into the software product. Further, an SQA plan did exist for this program and was jointly reviewed by the PM and contractor. The SQA plan outlined such items as the establishment of documentation standards, tools and support software necessary for testing purposes and all formal and informal reviews and audits which would be conducted through the course of the software development effort.

B. Independent support: A contracted IV&V team was used and initially was effective. However, as the program progressed and funding problems arose concerning the payment of the IV&V contractor, the IV&V team, in the opinion of the PM, became more of a hindrance than a help in software development. The contractor did have an independent SQA section whose actions were evident in daily software development operations.

C. Testing: All software was tested several times by the contractor prior to monitoring by AF engineers. The PM stated that the attitude of the contractor was one that strongly emphasized the early detection of errors.

D. Progression: All formal and informal reviews and audits were conducted to the satisfaction of the PM. Software development followed an orderly progression.

E. Other:

1. Software Metrics: Metrics were used by the IV&V team.
2. MIL-S-52779A: Not used.
3. The experience level of the PM combined with the type of contract appeared to further the quality of the software product produced.

MEETS BASELINE MANAGEMENT STANDARD: YES.

The functional baseline was established at contract award, the allocated baseline was established prior to PDR and the product baseline was established prior to CDR. Therefore, not only does this program meet the standard, but also exceeds its requirement by taking control of the product baseline sooner than need be.

Program 16

Contract Type:	FPI	Program Cost:	$8.75M
Software Cost:	$8.75M	Cost Overrun:	$0
Schedule Slip:	12 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: End user is adversely impacted

Realistic Budget: No
Realistic Schedule: No
Realistic System Specification: No

SQA CLASSIFICATION: UNSOUND.

A. Management support: With the exception of the PM, overt AF upper level management support did not exist for this program. However, the PM did have an extensive background in software development and appeared knowledgeable of the potential problems which could occur through the software development process. The contractor did have an upper level management policy, but the impact of this policy on the software development effort could not be determined. An SQA plan for this program did exist and was reviewed by Program Office personnel.

B. Independent support: A contracted IV&V team was used on this program and initially was effective. However, a problem which developed concerned product definition. Specifically, according to the PM, the user did not clearly define the product desired, and continuously requested design changes to keep pace with rapidly advancing technology. With so many changes occurring, the PM stated that the IV&V contractor became gradually frustrated and eventually ineffective. Consequently, the PM concluded that it was exceptionally difficult to maintain any semblance of an SQA program. The contractor did have an independent SQA section which was also initially classified as effective by the PM.

C. Testing: The contractors attitude initially was one of attempting to identify software discrepancies early in the software development process. However, with changes rapidly occurring in the desired product, this attitude evolved to one of "build then fix" the software.

D. Progression: The progression of the software developed was unsatisfactory in the opinion of the PM, and did not progress in an orderly manner.

E. Other:

1. Software Metrics: Metrics were used by the IV&V team.
2. MIL-S-52779A: Not used.
3. The problems that seemed to disrupt this effort were, first, the user did not understand what was needed, and, second, rapidly changing technology.

<u>MEETS BASELINE MANAGEMENT STANDARD: YES</u>.

This was an update program. Because specifications already existed, both the functional and allocated baselines were established at contract award. Thus, no preliminary design review was conducted since the development specifications was already under government control. The updated product baseline was established after CDR. Based on this description, the program is determined to have exceeded the Standard's requirements. Specifically, control of the allocated and product baselines was established far earlier than need be.

<u>Program 17</u>

Contract Type:	CPIF	Program Cost:	$121.1M
Software Cost:	$20M	Cost Overrun:	$2.8M
Schedule Slip:	12 months		

Significant Cost Overrun: 15%
Significant Schedule Slip: 4 months beyond delivery date

Realistic Budget: No
Realistic Schedule: No
Realistic System Specification: Yes

<u>SQA CLASSIFICATION: UNSOUND</u>.

A. Management support: Overtly, AF upper level management support did not exist. The contractor did have an upper level management support policy, but the effectiveness of this policy was questionable. An SQA plan existed, but it was only given a cursory review by Program Office personnel. The plan did outline such items as basic documentation requirements, as well as delineate all formal and informal reviews and audits. The Plan did not cover the tools and support software required for software testing.

B. Independent support: A contracted IV&V team was used and, in the opinion of the PM, was effective. The contractor did have an independent SQA function which the PM also believed actively contributed to the software product during the later stages of the software development.

C. Testing: The contractor initially appeared to have an attitude of "build then fix" the software. Initial contractor attitudes did not emphasize the early detection of software discrepancies.

D. Progression: The development of the software product initially did not proceed satisfactorily in the opinion of the PM. Both Preliminary Design Review and Critical Design Review were cancelled due to lack of contractor preparedness. However, after the PM began withholding progress payments, a more orderly progression of software development resulted.

E. Other:
1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. A single factor appeared to contribute to the development and eventual improvement of the software product developed and that factor was the PM's practice of withholding progress payments until work in a given phase of the development effort was satisfactorily completed.

MEETS BASELINE MANAGEMENT STANDARD: YES.

The functional baseline was established at contract award. The allocated and product baselines were established at PDR. The PDR was far more involved than required by MIL-STD-1521 or other guiding documentation. CDR information was included, thus, a product baseline was established at this time also. The program, thus, falls within the thesis' criteria for meeting the baseline management standard, and far exceeded it by establishing product baseline at PDR. This action was considered by the program manager to be severely unnecessary.

Program 18

Contract Type:	CPIF	Program Cost:	$141.1K
Software Cost:	$141.1K	Cost Overrun:	$0
Schedule Slip:	0 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: 5 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: AF upper level management emphasis did exist for this program. However, the contractor did not have any upper level management emphasis

concerning SQA. The PM was not aware of any SQA plan for this program.

B. Independent support: Neither a contracted nor a Program Office IV&V team was used on this program. The contractor did have an independent SQA function. However, this function was newly formed, and the PM discovered that the SQA section was in constant conflict with the engineering section which had previously accomplished all quality assurance responsibilities.

C. Testing: All software was tested several times prior to review by AF engineers.

D. Progression: Phases within the software development effort followed an orderly progression, which was to the satisfaction of the PM. Specifically, no new phase of development was undertaken until work in preceding phase had been satisfactorily accomplished.

E. Other

1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

3. Although problems were being encountered, the PM believed a quality software product was still being produced. The rationale behind this observation as made by the PM was that the company wanted to attempt to overcome previously bad publicity resulting from poor

company performance on previous government contracts.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The allocated baseline will not be established until delivery of software. Thus, the program does not meet the requirement of the baseline management standard.

Program 19

Contract Type:	CPIF	Program Cost:	$96M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	6 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: 4 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: No
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: The AF did not have any overt upper level management support to emphasize the significant costs and potential problems associated with software development. The contractor also did not have any upper level management emphasis. However, before the program started, the PM required the contractor to clearly state a position on software development. The contractor apparently was not agreeable to this but eventually complied. An SQA plan was developed for this program and reviewed by Program Office personnel. The plan outlined such items as documentation requirements and the formal and informal reviews and audits to be conducted through the course of the software development effort. Testing tools and support software were not outlined in the plan.

B. Independent support: A contracted IV&V team was used and was effective. The contractor did have an independent SQA function, but the PM classified this section as ineffective. Apparently, the contractor's SQA function had been newly formed and was in continuous state of conflict with the contractor's engineering function which had previously been responsible for quality assurance activities.

C. Testing: All testing was satisfactorily accomplished. The contractor's attitude emphasized the early detection of software discrepancies.

D. Progression: Before a new phase of software development was undertaken, work in preceding phase was accomplished to the satisfaction of the PM. Software development followed an orderly progression.

E. Other:
1. Software Metrics: Not used.
2. MIL-S-52779A: Used.

<u>MEETS BASELINE MANAGEMENT STANDARD: YES</u>.

The functional baseline was established at contract award, allocated baseline at PDR, and product baseline will be established at FCA. Therefore, the program meets the criteria of the baseline management standard.

Program 20

Contract Type:	FPIF	Program Cost:	$175M
Software Cost:	Data Unavailable	Cost Overrun:	$0
Schedule Slip:	7 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: 4 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: Overtly, AF upper level management support did not exist for this program. The contractor did have an upper management emphasis on SQA, but, as a result of scheduling pressures, the contractor could not adhere to SQA policy emphasis. An SQA plan was developed for this effort but was not reviewed by Program Office personnel.

B. Independent support: A contracted IV&V team was used for this software development effort. According to the PM, the IV&V team was effective. The contractor did not have an independent SQA activity. Further, in the opinion of the PM, the section responsible for SQA focused primarily on the hardware aspect of the program.

C. Testing: The contractor appeared to have an attitude of "build then fix" the software. The contractor did not emphasize the early detection of software discrepancies.

D. Progression: Many of the formal and informal reviews and audits had been conducted before the present PM was assigned to this project. Therefore, this portion of the analysis on this program was not evaluated.

E. Other:
1. Software Metrics: Not used.

2. MIL-S-52779A: Not used.

MEETS BASELINE MANAGEMENT STANDARD: YES .

The functional baseline was established at contract award, the allocated baseline was established at PDR, and the product baseline was established after CDR. Therefore, the program not only meets the requirements of the baseline management standard but exceeds its requirement by authenticating the product baseline earlier than need be.

Program 21

Contract Type:	FPI	Program Cost:	$80M
Software Cost:	$350K	Cost Overrun:	$0
Schedule Slip:	7 months		

Significant Cost Overrun: 10%
Significant Schedule Slip: 4 months beyond delivery date

Realistic Budget: Yes
Realistic Schedule: Yes
Realistic System Specification: Yes

SQA CLASSIFICATION: UNSOUND.

A. Management support: AF upper level management emphasis on SQA did exist; however, the extent to which such emphasis was effective was difficult to determine. No upper level management emphasis appeared to exist with the contractor. A SQA plan was not developed for this project.

B. Independent support: Neither a contracted nor a Program Office IV&V team was used on this software development effort. The contractor did have a separate SQA function, but the PM could not provide an assessment of the effectiveness of this section.

C. Testing: The PM stated that the software developed was not thoroughly tested prior to testing by AF engineers. The attitude of the contractor was one of 'build then fix' the software. The contractor did not emphasize the early detection of software discrepancies.

D. Progression: The PM stated that the formal and informal reviews and audits were not satisfactorily accomplished. Software development did not follow an orderly progression. Specifically, work in a new phase of the software development effort was often undertaken before work in preceding phases was satisfactorily accomplished.

E. Other:

1. Software Metrics: Not used.

2. MIL-S-52779A: Used.

MEETS BASELINE MANAGEMENT STANDARD: NO.

The allocated baseline was not established till FCA/PCA. Thus, the program fails to meet the criteria of the baseline management standard.

Bibliography

1. Air Force Systems Command. A Guide For Program Management. AFSCP 800-3. 9 April 1976.

2. Air Force Systems Command. Configuration Management. AFSCP 800-7. 1 December 1977.

3. Anway, Maj Mark D. "Configuration Management for the Development of Computer Systems." Unpublished report No. PMC 76-2. Fort Belvoir VA, November 1976.

4. Babel, Philip S. ASD Computer Resource Focal Point. Personal interview. ASD, Deputy for Engineering, Wright-Patterson AFB OH, 4 May 1984.

5. Barr, Lt Noah. ASD, Software Quality Assurance Focal Point. Personal interview. ASD, Deputy for Production and Manufacturing, Wright-Patterson AFB OH, 10 May 1984.

6. Black, Rachael K. E., an' others. "BCS Software Production Data." Unpub. ied report No. RADC-TR-77-166, Rome Air Development Center NY, March 1977.

7. Carlson, Lt Col Robert D., Software Acquisition Manager. Personal interview. ASD, Deputy for Simulators, Wright-Patterson AFB OH, 10 May 1984.

8. Chow, Tsun S. "Advances in Software Quality Assurance," Proceedings of the Computer Society's Sixth International Computer Software and Applications Conference. 82CH1810-1. IEEE Computer Society Press, Silver Spring NY, 1982.

9. Cook, M.L. "Software Metrics: An Introouction and Annotated Bibliography," SIGSOFT Vol. 7, No. 2: Pg 41-60, (April 1982).

10. Cooper, John D. and Matthew J. Fisher. Software Quality Management. New York: Petrocelli Books, Inc., 1979.

11. Daneman, Jeffery. Assistant Professor of Quantitative Methods. Personal interview. AFIT/LSQ, Air Force Institute of Technology, Wright-Patterson AFB OH, 11 July 1984.

12. de Boer, U. Posthuma. "Cost Estimation and Management Control of Software Development in Scientific/Technical Projects." Unpublished report No. NLR-TR-78056-U, National Aerospace Laboratory NLR The Netherlands, May 1978.

13. DeMarco, Tom. Concise Notes on Software Engineering. New York: Yourdon Press Monograph, 1979.

14. Denicoff, M. and R. Grafton. "Software Metrics: Paradigms and Processes," Computer Science and Statistics: Proceedings of the 13th Annual Symposium on the Interface. Springer Verlag NY 1981.

15. Department of Defense. Configuration Management. DoDD 5010.19. Washington: Government Printing Office, 17 July 1968.

16. Department of Defense. Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs. Mil-Std 483. Washington: Government Printing Office, 31 December 1970.

17. Department of Defense. Quality Assurance. DoDD 4155.1 (Enclosure 2). Washington: Government Printing Office, 1972.

18. Department of Defense. Technical Reviews and Audits for Systems, Equipments, and Computer Programs. Mil-Std 1521. Washington: HQ USAF, 29 September 1978.

19. Department of the Air Force. Configuration Management. AFR 65-3. Washington: HQ USAF. 1 July 1974.

20. Department of the Air Force Regulation. Acquisition and Support Procedures for Computer Resources in Systems. AFR 800-14, Vol II. Washington: HQ USAF, 26 September 1975.

21. Driscoll, Lt Col Alan J. "Software Visibility and the Program Man ger," Defense Systems Management Review, Vol I: 12-27, Spring 1977).

22. Dunn, Robert, and Ullman, Richard. Quality Assurance for Computer Software. New York: McGraw-Hill Book Company, 1982.

23. Elliot, Maj Don A. "A Guide to Configuration Management for Army Managers of Software R&D Projects." Unpublished report. U.S. Army Command and General Staff College, Fort Leavenworth KS, April 1976.

24. Emory, William C. Business Research Methods. Homewood: Richard D. Irwin, Inc., 1980.

25. Faidell, George F. "Software Quality Assurance," 1983 Conference Proceedings Second Annual Phoenix Conference. 83CH1864-8. IEEE Computer Society Press, Silver Spring NY, 1983.

26. Freeman, Peter and Anthony I. Wasserman. Tutorial on Software Design Techniques. IEEE, Cat. no. 76CH114s-2C, 2nd ed., 1977.

27. Gansler, Jacques S. "Comment," Defense Management Journal. Vol II: 1, (October 1975).

28. Gilb, Thomas. Software Metrics. Cambridge: Winthrop Publishers, Inc., 1977.

29. Jarzcmbek, Stanley J., Jr. Software Quality Metrics: A Software Management Monitoring Method for AFLC in Its Software Quality Assurance Program for the Quantitative Assessment of the System Development Life Cycle Under Configuration Management. MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1982. (AD-A115 501).

30. Johnson, Marshall S. "The Role of Software Quality Measurement in Software Development," The IEEE Computer Society's Sixth International Computer Software and Applications Conference. 82CH1810-1. IEEE Computer Society Press, Silver Spring NY, 1982.

31. Lauber, R.J. "Impact of a Computer Aided Development Support System on Software Quality," The IEEE Computer Society's Sixth International Computer Software and Applications Conference. 82CH1810-1. IEEE Computer Society Press, Silver Spring NY, 1982.

32. Lecture materials distributed in SYS 201, Technical Administration of Embedded Computer Resource. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, undated.

33. Lloyd, Lewis E. Techniques for Efficient Research. New York: Chemical Publishing Co., 1966.

34. Martin, Edith W. "Strategy for a DoD Software Initiative," Computer, Vol 16: 52-59, (March 1983).

35. McCall, J., P. Richards, and G. Walters. "Factors in Software Quality." Unpublished report No. NITS AD-A049 014, AD-A049 015, AD-A049 055, Rome Air Development Center NY, November, 1978.

36. McClave, James T. and P. George Benson. Statistics for Business and Economics. Santa Clara: Dellen Publishing Co., 1982.

37. Meinke, George H. "Airborne Software Acquisition Management: A Guide for New Software Managers." Unpublished report No. ACSC-82-1685, Air War College, Maxwell AFB AL, April 1982.

38. Miller, Edward F., Jr. and William E. Howden. Tutorial: Software Testing and Validation Techniques. IEEE Cat. no. 138-8, 1978.

39. Murine, Gerald E. "The Application of Software Quality Metrics." 1983 Conference Proceedings Second Annual Phoenix Conference. 83CH1864-8. IEEE Computer Society Press, Silver Spring NY, 1983.

40. Nelson, Gary J. "Putting Computer Software to the Test," Defense Management Journal, Vol 15: 38-41, (March-April 1979).

41. Patterson, Alton E. "Embedded Computer System Software Management Support After System Deployment." Unpublished report No. PMC 77-1, Defense Systems Management College, Fort Belvoir VA, May 1977.

42. Perlis, Alan, Frederick Sayward and Mary Shaw, eds. Software Metrics: An Analysis and Evaluation. Cambridge: The MIT Press, 1981.

43. Poston, Robert M. "Software Quality Assurance Implementation," The IEEE Computer Society's Sixth International Computer Software and Applications Conference. 82CH1810-1. IEEE Computer Society Press, Silver Spring NY, 1982.

44. Riefer, Donald J. "Tutorial: Software Management," IEEE Catalog No EHO 146-1, 1979.

45. Ruby, Raymond J. and R. Dean Hartwick. "Quantitative Measurement of Program Quality," Proceedings of the 1968 ACM National Conference. August 1968.

46. Thorndike, Robert L. and Elizabeth Hagen. Measurement and Evaluation in Psychology and Education. New York: John Wiley & Sons, Inc., 1969.

47. Wade, Gerald O. Responsiveness and Configuration Control for Embedded Computer Software. MS thesis. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1980. (AD-A088 723).

48. Whited, Jon A. "Management Control Practices for Software Quality," Software Quality Management. New York: Petrocelli Books, Inc., 1979.

VITAE

Captain Sidney C. Kimhan III was born on 31 May 1956 in Stutgardt, Germany. He graduated from high school in Kailua, Hawaii, in 1974, and attended the University of Hawaii from which he received the degree of Bachelor of Science in Accounting in August, 1978. Captain Kimhan received his commission in the USAF through OTS. He was assigned to Aeronautical Systems Division in the Deputy for Aeronautical Equipment as a program manager in charge the Depot Automatic Test Equipment acquisition program until entering the School of Systems and Logistics, Air Force Institute of Technology, in May, 1983.

Permanent address: 620-B Oneawa Street
Kailua HI 96734

Captain David M. King was born on 28 June 1957 in Cambridge, Ohio. He graduated from high school in Reading, Massachusetts, in 1975, and attended the United States Air Force Academy from which he received a Bachelor of Science degree in May, 1979. Upon graduation, he received a commission in the USAF and was assigned as the Base Fuels Management Officer at RAF Upper Heyford, England. During this time, he also attended Vanderbilt University's Overseas Extension program, receiving a Master of Science degree in Education. Reassigned to Pease AFB, New Hampshire in September of 1981, he served as both the Customer Support and Material Management Officer in base supply until May, 1983.

Permanent address: Appartdo 53-983
Mexico DF 17
Mexico

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GLM/LSM/84S-35	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Systems and Logistics; 6b. OFFICE SYMBOL *(If applicable)* AFIT/LS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.
11. TITLE *(Include Security Classification)* See Box 19	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

12. PERSONAL AUTHOR(S)
Sidney C. Kimhan III, Capt, USAF   David M. King, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1984 September	142

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
[illegible] 14 Sept 84
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Software Quality Assurance, Software Baseline Management, Software Acquisition, Software Development, Computer Programs, Software Metrics

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: THE EFFECT OF SOFTWARE QUALITY CONTROL AND BASELINE MANAGEMENT ON THE ACQUISITION OF COMPUTER PROGRAMS

Thesis Advisor: Mr. William Dean, Associate Professor

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mr. William Dean, Associate Professor	22b. TELEPHONE NUMBER *(Include Area Code)* 513-255-3355; 22c. OFFICE SYMBOL AFIT/LSY

Cost effective development of quality software for new system acquisition is an issue of increasing concern within the Department of Defense. This thesis examines the issue of software development for Embedded Computer Systems, within the context of the Software Development Life Cycle (SDLC), from the perspectives of Software Quality Assurance (SQA) and Baseline Management. The objective of this research effort is to develop an approach for the joint application of SQA and Baseline Management to improve management control (maintain cost and schedule integrity) during the software development process.

Initially, the disciplines of SQA and Baseline Management are presented as individual components, operating during the SDLC, which provide management with increased control over the software development process. General concepts associated with SQA are first discussed, including the potential role of Software Metrics. This is followed by a review of Department of Defense literature pertaining to SQA and Baseline Management, which are then studied as a combined approach towards the effective management control of the software development process.

Through the use of a structured interview, twenty-one Program Managers were surveyed. Form the collected cost, schedule and program history data, programs were classified as either having a "Sound" or "Unsound" SQA program and as either "adhering to" or "not adhering to" a baseline management standard. Consequently, analyzing the data using Student's t statistic, the major finding of this research is that there is no statistical evidence that the application of either a "Sound" SQA program or baseline management standard results in positive cost and schedule control.

ATE
LME